AFRL-RX-WP-TR-2012-0359

INSTANT FOUNDRY ADAPTIVE THROUGH BITS (iFAB)

Tolga Kurtoglu
Palo Alto Research Center, Inc.

Matthew Campbell
University of Texas at Austin

Joseph Rasche
Rolls Royce

Jami Shah
Arizona State University

Yorgos Stilyanos
Mission Critical Technologies

JULY 2012
Final Report

AIR FORCE RESEARCH LABORATORY
MATERIALS AND MANUFACTURING DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7750
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the USAF 88th Air Base Wing (88 ABW) Public Affairs Office (PAO) and is available to the general public, including foreign nationals.

Qualified requestors may obtain copies of this report from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RX-WP-TR-2012-0359 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//SIGNED//	**//SIGNED//**
STEVEN A. TUREK, Program Manager	SCOTT M. PEARL, Branch Chief
AFRL/RXMS	AFRL/RXMS

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
July 2012	Final	2 May 2011 – 31 May 2012

4. TITLE AND SUBTITLE
INSTANT FOUNDRY ADAPTIVE THROUGH BITS (iFAB)

5a. CONTRACT NUMBER
FA8650-11-C-7130

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62303E

6. AUTHOR(S)
Tolga Kurtoglu (Palo Alto Research Center, Inc.)
Matthew Campbell (University of Texas at Austin)
Joseph Rasche (Rolls Royce)
Jami Shah (Arizona State University)
Yorgos Stilyanos (Mission Critical Technologies)

5d. PROJECT NUMBER
3000

5e. TASK NUMBER
00

5f. WORK UNIT NUMBER
M0129900

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Palo Alto Research Center, Inc.
3333 Coyote Hill Road
Palo Alto, CA 94304-1314

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
Materials and Manufacturing Directorate
Wright-Patterson Air Force Base, OH 45433-7750
Air Force Materiel Command
United States Air Force

Defense Advanced Research Projects Agency/ (DARPA)
3701 N. Fairfax Drive
Arlington, VA 22203-1714

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/RXMS

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-RX-WP-TR-2012-0359

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES
This work was funded in whole or in part by Department of Air Force contract FA8650-11-C-7130. The U.S. Government has for itself and others acting on its behalf an unlimited, paid-up, nonexclusive, irrevocable worldwide license to use, modify, reproduce, release, perform, display, or disclose the work by or on behalf of the U.S. Government. PA Case Number and clearance date: 88ABW-2012-4091, 20 July 2012. This document contains color.

14. ABSTRACT
The goal of this project is to incorporate manufacturability constraints early into the system design process to enable design trade space exploration which we believe will shorten schedules, decrease manufacturing costs, increase reliability and enable reconfiguration and reuse capability within design and manufacturing processes. Our long-term vision is to embed the proposed intelligent design and manufacturing framework into existing product lifecycle management software technologies so that corporations can dramatically improve product development time and reduce product lifecycle management costs while capitalizing on concurrent engineering practices.

15. SUBJECT TERMS
Instant Foundry Adaptive through Bits (iFAB), manufacturing model library, manufacturing, processes, foundry

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Steven A. Turek
Unclassified	Unclassified	Unclassified	SAR	74	19b. TELEPHONE NUMBER *(Include Area Code)* N/A

Table of Contents

Section **Page**

List of Figures

Figure **Page**

1.0 SUMMARY

1.1 Overview

The work reported herein was performed by the Palo Alto Research Center, Incorporated under Air Force Research Laboratory (AFRL) Contract FA8650-11-C-7130, "iFAB Technical Area III: Manufacturability Constraint Feedback to System Design, "Reasoning about Manufacturability and Design Changes via Graph-Grammar Based Search". The Period of Performance (PoP) of the effort was 02 May 2011 to 31 May 2012. The DARPA Technical Program Manager was initially Paul Eremenko and as of 21 February 2012 was Nathan Wiedenman. The PARC Program Manager was initially Kelly Coupe and as of 01 January 2012 was Nora Boettcher; Dr. Tolga Kurtoglu was Principal Investigator (PI) with Dr. Christian Fritz. Subcontractors were University of Texas at Austin, with lead Matthew Campbell; Rolls Royce LibertyWorks with lead Mr. Joseph Rasche; Mission Critical Technologies with Yorgos Stilyanos; and Arizona State University with lead Dr. Jami Shah.

1.2 Goal

The goal of this project is to incorporate manufacturability constraints early into the system design process to enable design trade space exploration which we believe will shorten schedules, decrease manufacturing costs, increase reliability and enable reconfiguration and reuse capability within design and manufacturing processes. Our long-term vision is to embed the proposed intelligent design and manufacturing framework into existing product lifecycle management software technologies so that corporations can dramatically improve product development time and reduce product lifecycle management costs while capitalizing on concurrent engineering practices.

1.3 Summary of Approach

As part of the iFAB program, the PARC led team developed and demonstrated a novel software tool for automatically determining manufacturability of a given part geometry based on the process capabilities of a specific foundry. The AMFA tool (Automated Manufacturability Feedback Analysis) takes as input a CAD (Computer-Aided Design) file, analyzes the geometry to determine whether the part can be manufactured from existing raw material, and if so, outputs one or more detailed process plans.

The AMFA tool streamlines the interaction between the design process and manufacturing as shown in Figure 1. In Figure 1, a feedback loop is shown where designers submit the technical data package to manufacturing so that the design may be physically constructed. AMFA then analyzes the CAD files within the technical data package, which are to be constructed (as opposed to those provided by OEMs). In order to facilitate open directives and to be agnostic relative to what CAD software will be used in the design process, the CAD files are requested as STEP files, which is one of the most common CAD data exchange formats in use. AMFA then performs an analysis of the STEP file by referencing the data provided on the manufacturing facility that will build the part in question.

The outputs of AMFA are a prediction for the time and cost to create the part, which the user can immediately use to consider if redesign is necessary. In addition to these values, a general fabrication plan is supplied along with a view of the part in between models (also known as staging models as shown in Figure 1). Finally, in certain cases, changes are recommended to the designer if the part in question is non-manufacturable or is manufacturable at a high cost.

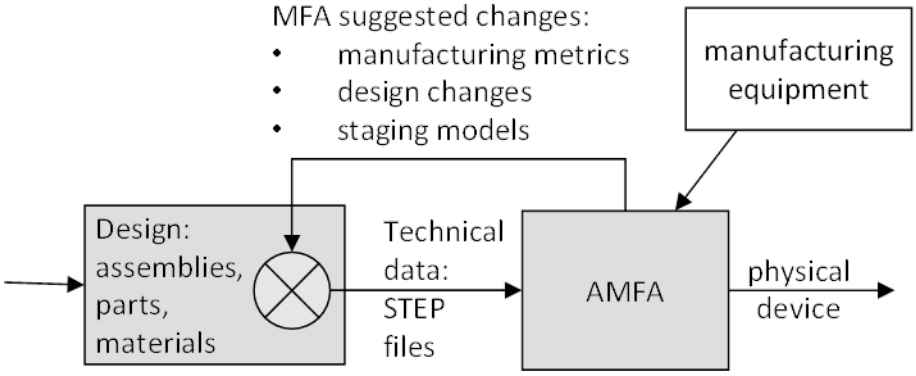


Figure 1. AMFA Supports Manufacturability Feedback Analysis

In order to provide this output, a simulation is performed which reasons about the computational geometry. The goal is to provide feedback to the designer in real-time so that they can immediately consider changes to the part. In the current implementation, AMFA runs within a web browser on a cloud architecture. Figure 2 shows a screenshot of the tool. In the upper corner, there is a small button to upload a single STEP file. For example, this data may be prepared in the background by scouring the technical data package for CAD files for custom parts. Within the four frames shown, data is provided back to the user on the features found within the part and the possible changes that may be pursued (Figure 2 (a)), a Pareto plot of various alternate ways to construct the part (axes are time vs. cost in Figure 2 (b)), a detailed plan for one chosen alternative (Figure 2 (c)), and a view of the stages of the part (Figure 2 (d)).

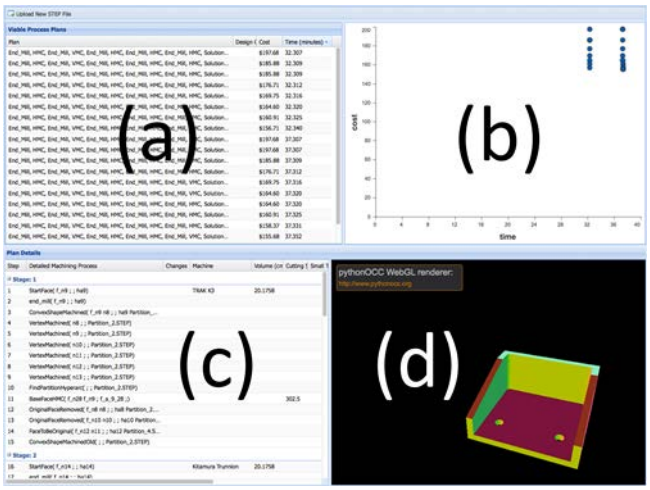


Figure 2. A Screenshot of the AMFA System

The main research and development focus is not on this data visualization but rather the underlying simulation that creates the data. In the past year, the team applied a variety of computational innovations to define this simulation: computational geometry, graph grammars, AI tree search and planning. After a STEP file is loaded, the geometry – comprised of vertices, edges, and faces – is parsed into a label-rich graph of nodes, arcs, and hyperarcs which serves as the basis for the search and simulation. In the translation, a bounding box is extrapolated from the part shape since one would likely start the actual fabrication for a larger block of material. The positive part shape is then subtracted from the bounding box to create the negative volume that is to be removed. This negative volume then undergoes further division. In the current efforts, the bounding box subtraction allows for the reasoning about subtractive machining operations such as milling and drilling.

The resulting graph of positive and negative elements serves as a seed in a tree-search. Steps in the tree represent alternative manufacturing operations. These operations are determined through grammar rules which detect a series of graph elements and relate them to a particular manufacturing process. As the tree grows, many alternative plans are derived and evaluated on the time-vs.-cost scale. The time is comprised of five distinct types: productive machining time, machine switch times, tool switch times, fixturing and unfixturing times, and deburring and inspection times. Cost is comprised of tooling cost, machine amortization cost, and operational costs. The calculations of these are based on data from the machine database and details are stored in the graph representing the shape.

AMFA generates plans which consist of: machine choices, tool choices, part orientations, and machining speeds. The method stops short of defining tool paths which could result in actual CNC code generation. Essentially, there are various computer-aided manufacturing software tools which create detailed path information. In those methods, initial conditions are requested in the form of the very decisions that AMFA is searching for. Therefore, AMFA is searching amongst high-level planning decisions with a plan to offload detailed instruction creation to other tools.

In the current AMFA system, feedback is provided to the designer in one of two ways:

Compliance with generic Design for Manufacturing (DFM) principles: This is implemented as geometric constraints formulated as DFM rules. These rules tend to be specific to a high-level manufacturing process, in this case for milling and drilling. Before exploring possible process plans, a submitted part is first tested for violations of any DFM rules. The results are presented graphically in the GUI by highlighting issues that raised flags in particular DFM tests. The designer can then modify their part accordingly and resubmit.

Relaxation search: If the search for a process plan comes back negative, i.e., no plan could be found, then a secondary search, called relaxation search is performed. In this search, changes that could be made to the non-manufacturable design that will make it manufacturable are identified. Intuitively, this search takes liberties as it explores the possibilities making the design manufacturable. Relaxation search aims to find the minimal set of changes to the part geometry, in order to still find a fabrication solution.

2.0 INTRODUCTION

The design and manufacture of cyber-physical systems, systems with ever increasing functional complexity and shorter system lifecycles, presents many unique challenges. One of the challenges is the lack of coupling between the conceptual design phase and the manufacturing phase. Today these coupling issues are typically resolved through multiple design-build-test-redesign iterations leading to longer schedules, capital-intensive manufacturing costs, reduced reliability and limited reconfiguration and reuse capability within the manufacturing enterprise.

To address these challenges, The DARPA AVM (Defense Advanced Research Projects Agency Adaptive Vehicle Make) portfolio of programs is pursuing revolutionary approaches to the design, verification, and manufacture of complex defense systems and vehicles. The AVM portfolio includes the C2M2L, META, FANG and iFAB programs. The iFAB program (Instant Foundry Adaptive through Bits) is laying the groundwork for the development of a foundry-style manufacturing (FSM) capability—taking as input a verified system design specified in an appropriate meta-language (i.e. the output from the META program)—capable of rapid reconfiguration to accommodate a wide range of design variability and specifically targeted at the fabrication of military ground vehicles. The principal objective of iFAB is to enable substantial compression of the time required to go from idea to product through a shift in the product value chain for defense systems. The iFAB vision is to move away from a capital-intensive manufacturing facility around a single defense product, and toward the creation of a flexible, programmable, potentially distributed production capability able to accommodate a wide range of systems and system variants with extremely rapid reconfiguration timescales. The specific goals of the iFAB program are to rapidly design and configure manufacturing capabilities to support the foundry-style manufacturing (FSM) of a wide array of infantry fighting vehicle models and variants.

Successful implementation of foundry-style manufacturing (FSM) requires information transfer between design representations (AVM META) and manufacturing representations (AVM iFAB) to be concurrent, automatic and efficient. The difficulty in achieving this is the management of the vast amount of heterogeneous data, knowledge and complicated inference processes encountered in developing intelligent manufacturing systems. Collaborative design and knowledge reuse in engineering product design and manufacturing becomes critical for the success of FSM and requires computational frameworks that effectively support representation, retrieval, and reuse of design and manufacturing information. One technology to enable an FSM facility as envisioned in iFAB is a suite of design tools to communicate what can and cannot be fabricated to the designers. For the AVM program, this suite of design tools fills an important gap in formulating and translating the aforementioned data, knowledge, and information between the design and manufacturing disciplines.

The iFAB team from PARC is developing this suite of automated manufacturability feedback analysis tools, named AMFA, to address this important gap. AMFA tools take as input a given foundry configuration along with the specification of part shapes to automatically conduct trade-space analyses pertaining to machine, tool, and process selection. Thus, the designer can study the effect of changes to both the design and the foundry to determine manufacturability of a part. In this way, it mimics the machinist expertise that understands the intricacies of the fabrication

machines, the proper ordering of operations, and the ways to subtly change the design to make a non-manufacturable part manufacturable or to reduce manufacturing time and cost. In subsequent sections, we further describe the details of the approach we have taken that resulted in successful deployment of the aforementioned AMFA tool suite.

3.0 METHODS, ASSUMPTIONS AND PROCEDURES

In our approach we cast the problem of determining manufacturability and providing detailed feedback as a search problem over possible process plans that can be executed in a given foundry to produce the desired geometry.

To deal with the continuous, three-dimensional nature of the problem space, our approach starts by performing a detailed geometric analysis of the shape and producing an intermediate, discrete representation of the geometry. This step, described in Section 3.1, computes the negative volume that remains after subtracting the desired geometry from the chosen bar stock -- usually the bounding box. In this step, the negative volume is decomposed into partitions that can be removed/machined in one operation, i.e., one setup.

Given these partitions, a search is performed in a forward direction starting with the negative volume. In this search the intermediate representation of the geometry is used to capture the intermediate stages the geometry goes through as the hypothesized process plans are developed in the search. The goal is to find a state where all partitions have been marked as having been machined.

If process plans are found, they are presented in terms of a tradeoff space of metric attributes like cost, time, and manufacturing quality. This is to enable the designer to easily consider the advantages and drawbacks of various optimized solutions. If no plan is found, the design is proven to be non-manufacturable by the current foundry. In this case a relaxation search is performed. This relaxation search explores the space of possible design changes that would make the design manufacturable in the current foundry.

3.1 Geometric Representation and Reasoning

A solid object consists of elements (vertices, edges and faces) that are inter-connected. Given this fact, we can model any solid in a form of graph representation. This, however, may result in a loss of information while converting a solid to graph which needs be taken into consideration and depending on the application, one may add as sufficient information as needed to the graph structure. We will discuss our method of converting solid to graph which then will be used as a seed (input) for the graph grammar. In the past literature, few graph representations have been developed for solids models and each of them are tuned for specific purposes. In fact, there is no need to define a generic graph representation for solid objects and each application may need its own version of seed graph. To reason about manufacturability we need certain elements from the solid to map into a graph and we also need additional information for each element such as neighboring data, relative positioning, attributes and etc. In order to extract required information from a solid object and perform reasoning about the geometry we have used Open CASCADE API [1] as our open-source geometric kernel. Kernel functionality enables us to read and reason the solid model and translate data to and from the graph. For the input CAD format we use STEP. In general, there are numerous possibilities in CAD formats where few of the most common ones are explored in Section 3.1.1.

3.1.1 Exploring CAD Formats

In CAD data exchange there are several methods and software technologies involved to properly translate a model from one computer-aided design system to another CAD file format. Besides the geometrical (curve, wireframe, surface and solid) information there are other important data which needs to be translated to graph such as attributes, metadata, feature data and etc. One way of transferring CAD models between different systems is using a common intermediary format, where, each CAD system can read or write to it. Some CAD formats are independent of commercial systems and some others are established by standard organizations. A number of very common and widely used formats are various STEP standards such as, STEP AP203, STEP AP210, STEP AP212, STEP-NC AP238 and STEP AP242. In this research work we use STEP AP203, this format - together with STEP AP214 - is among the most commonly used in many mechanical CAD systems.

3.1.2 Graph Development

As mentioned above, one of the most important challenges in converting a CAD model is the chance of losing information while translating. Moreover, each system may have some internal data loss in reading or interpreting data from/to a different CAD system. Thus, it is essential to identify what is important and needed in a specific application to be included in the graph structure. In this section we will describe our method of translating solid to the native graph XML format (called GXML where G stands for Graphic purpose and XML is Extensible Markup Language). For geometric reasoning and manufacture process planning applications, only 3D geometry is needed to map to the seed graph. This is equivalent to model descriptions with some level of details including dimensions and topology data (boundary representation abbreviated as B-Rep). Other information such as feature data, assembly structure, texts, fonts and attributes (such as colors and graphic data) are not required at this point.

Two efforts are involved in the representation scheme. They are summarized into the Convex Decomposition and the Grammar Representation. In Section 3.1.3, a new method of decomposing 3D solid models is introduced. To begin with, a CAD model in STEP format AP203 is provided by designers: this geometry – comprised of vertices, edges, and faces – is decomposed into smaller convex shapes which serve as the basis for the representation rules. In the decomposition, a bounding box is extrapolated from the original part (input is a CAD model also referred to as a STEP file) since one needs to start the actual manufacturing from a larger block of material. The original part is then subtracted from the bounding box to create the volume that is to be removed Figure 3 (b). This removal volume then (the bounding box or initial work-piece Figure 3 (c)) is generated with the help of a geometric kernel (Open CASCADE Technology, 3D modeling & numerical simulation [1]). The material to be removed is called the negative solid and is obtained by subtracting the positive solid from its bounding box Figure 3(b). This Boolean subtraction is commonly referred to as disjunction in computational geometry. The goal is then to decompose the negative solid into smaller sub-convex shapes which are the bases of further analysis in AMFA.

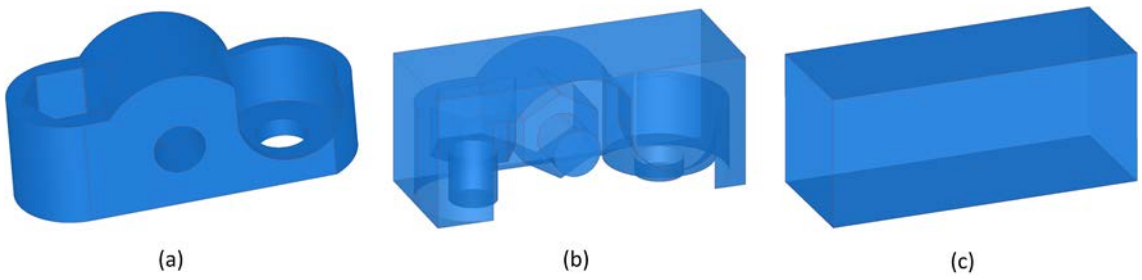


Figure 3. Illustrative Sample Part

3.1.3 Problem Formulation

The work is extended beyond the volumetric decomposition approach for 3D solid models by adding a layer of reasoning to the algorithm. The convex decomposition is a special case of a more general form-feature-recognition technique which is called machining-feature-generation where all generated convex shapes belong to a negative solid and are subtracted from the bounding box or work-piece. Our decomposition algorithm uses two distinct sets of half spaces to decompose a solid into pieces. The first type of half space is derived from planar faces which are attached to a concave edge (shown in Figure 4 (a,b)). The normal of the faces adjacent to the concave edge suggest two cutting planes for slicing the solid. In order to generate this type of half space, a straight concave edge is first detected by the algorithm. Two faces connecting to this concave edge can generate two face-normals and hence two cutting planes to split the solid into left and right volumes (Figure 4 (a, b)). The second type of half-space is derived from a cylindrical surface that is formed from an arbitrary planar curved edge (Figure 4 (d, e)). In this case, two distinct sets of cutting surfaces are defined. The first set is the plane defined by the curved edge, which is independent of the curvature (Figure 4 (c)). Therefore the cutting plane and the curve will always lie in the same plane. On the other hand the second set is dependent on the curvature of the curve and whether it is part of a circular edge or not. For the latter case, a cylindrical cutting surface with radius equal to the curvature of the curve is utilized. For all complete/closed circular edges a semi-infinite cylindrical surface is used as the cutting half-space, and finally, for all open circles we use an infinite cylinder as shown in Figure 4 (d, e). Therefore, generally speaking, for each concave edge either straight or curved there are two options for cutting the solid. Each option generates at least two distinct solids (because each cut-side contains at least one solid). It is important to note that each solid may not possess a pure convex shape. In other words, a convex shape may contain a curved concavity but these types of concavities are not considered for further divisions in this work because, in principle, they can generate infinitely many convex pieces which are not suitable for process planning purposes. The total number of solutions for a shape with n number of concave edges is equal or smaller than 2^n and is discussed in more detail in 3.1.6.

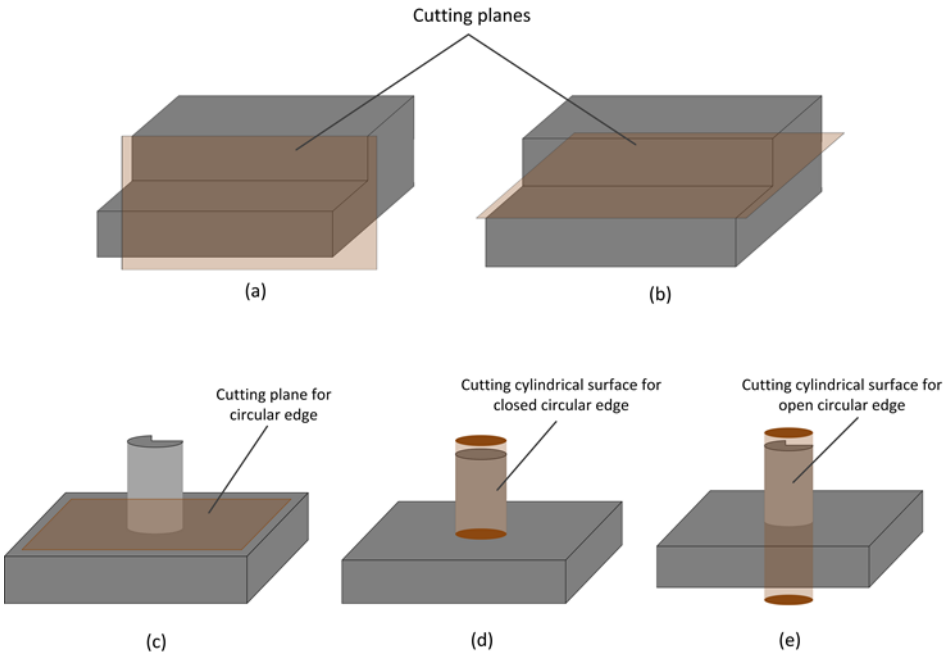


Figure 4: Different Cutting Surfaces Used in Convex Decomposition Algorithm

3.1.4 Background

In the context of computer-aided manufacturing, convex decomposition is important and useful for generating simple removal volumes from the work-piece. These topological entities are commonly referred to as machining features. In theory, a decomposed solid can be represented as the sum of sub-convex-volumes in a categorized order that forms a complete object obtained from its boundary representation (B-rep). Different features intrinsic to the shape of an object can be recognized using the convex-decomposition approach. One negative aspect of this method is the possibility of adding complexity to the initial model through the decomposition, this may occur because, feature generation can become more and more intricate and computationally slow as the number of convex regions grows. In order to prevent this problem and yet to have a powerful decomposition algorithm, a novel partitioning scenario that uses two distinct types of half-spaces for cutting the solid was implemented. It includes a concave-edge ranking strategy to prioritize cuts and two sets of heuristics to evaluate each cut. The proposed framework is useful for recognizing machining steps as indicated by addressing sections of the negative volume.

3.1.5 Convex Decomposition

AMFA incorporates a new method of decomposing 3D solid models for use in automated manufacturing process planning applications. The algorithm is based on dividing complex volumes into convex sub-volumes. In order to ensure that the resulting convex sub-volumes are compact and feasible for machining operations, a set of heuristics were developed to categorize, order, and determine splitting directions of the concave edges. Each of the resulting convex sub-volumes represents material that is to be removed from the bounding-box to create the intended

work-piece. The set of sub-volumes represents a compound solid (i.e. decomposed solid) that is converted into graph of boundary representation (B-rep) primitives and is reasoned about in a subsequent search process. The algorithm was implemented and tested on a variety of solid models resembling real automotive parts, and results show the effectiveness and efficiency of this algorithm.

3.1.6 Exploring Solutions

Convex decomposition results can be represented as a tree structure with branching factor equal or greater than 4 (4 is the case when there are exactly two solids generated after each cut) and depth of the tree equal to the total number of concave edges in the solid (Figure 5). Each node represents a volume that needs to be cut and each arc represents a left (L) or right (R) cut in the tree. Nodes can consist of simple shapes (i.e. a convex volume) which are represented as (S) in the tree or complex shapes (i.e. a volume with one or more concavities) which is represented as (C). As shown in Figure 5, the original solid is on top of the tree with a node containing letter C which denotes that it is a complex volume and needs to be decomposed. Convex decomposition starts by selecting a concave edge from the original solid model and n is the total number of concave edges in the original solid, but the actual amount is always smaller than this value because many branches stop after hitting a (S) node or simple volume. This is explained more in the following sections.

For simplicity, let us consider a case where the left branch is chosen as the preferred cut, and the result is one complex (C) volume and one simple (S) volume. The simple volume does not need any further cutting operations so the branch related to this node stops here; this is shown as a red node in the tree. For each complex volume (C) the branch grows further to lower levels until it hits a simple volume (S). In order to find the best result (or decomposed solid) one can generate all the possible options in the tree and explore and evaluate each individual option from the pool of solutions. This however is not a good idea due to the fact that Boolean operations are computationally expensive and slow in CAD kernels. Furthermore evaluation conceptually requires a human to inspect the result and decide if it is a good decomposition or not (computationally evaluating the quality may be possible, but it is out of the scope of this work). Therefore, it is nearly impossible to explore all the options or branches in the tree. The alternative solution is to expand a single but promising branch. At each level in the tree the algorithm evaluates the solutions and decides the preferred direction for the next level. This continues until no more complex volumes are detected. In order to reach this goal, two sets of heuristics are designed and implemented to guide the algorithm in the desired direction.

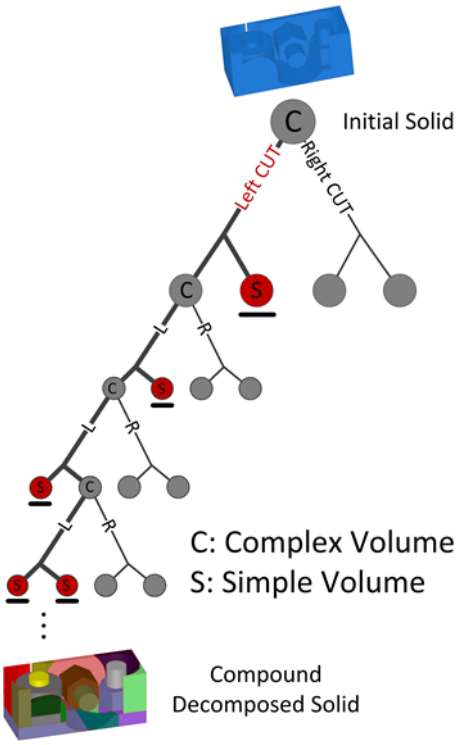


Figure 5. Convex Decomposition Tree

It is important to note that a desired solution is described as a decomposed volume that contains partitions which are suitable for manufacturing purposes. In other words, all the sub-convex-volumes should possess certain properties which include (1) having a compact shape with no or few number of concavities, (2) having a prismatic or close to prismatic geometry. In order to accomplish these requirements the flowchart of Figure 6 was developed.

3.1.7 Selecting Candidate Edge for Cut

The concave edges are classified and ordered to meet the requirements specified above and are described as following. The first sets of concave edges are full (with 360°degree angle) or semi (with 180° degree angle) circular edges. These cutting edges lead to either cylindrical cutting surfaces or planar cutting planes where both can extract all the cylindrical features (such as holes in positive solid) from the solid model and are shown as edge types 1 & 2 in flowchart of Figure 6.

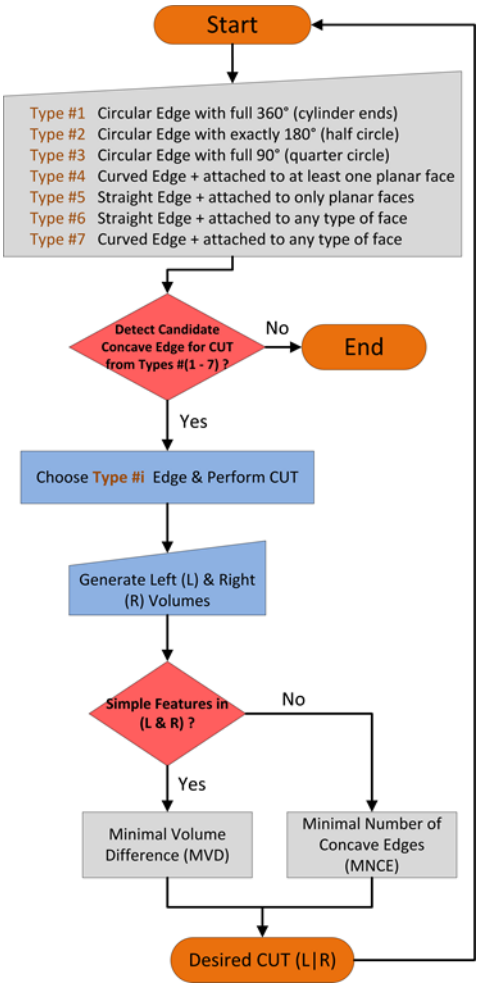


Figure 6: Convex Decomposition Flowchart

The second set of cutting edges is quarter circular edges (90° degree angle). These types of concave edges are primarily responsible for extracting fillets (or fillet-like features) from the solid model and are shown as type 3 in the flowchart.

All the curved concave edges that have at least one planar face attached to them are selected next. Based on our experiment and investigation in various kinds of mechanical parts, the reason for this selection is that extracting and removing features associated with these types of concavities prior to other types can simplify the solid geometry faster and in a more efficient manner. These edges are shown as type 4 in the flowchart.

Next in the hierarchy (type 5), are the straight concave edges which are attached to only planar faces. This condition, having attached to only planar faces, alleviates the chances of destroying the solid geometry and eliminating simple features. This is because these features generally contain planar faces and the condition ensures that forthcoming cuts could not affect simple partitions when they are removed a priori.

The cutting edges in 6th category are similar to those in type 5 but have no condition on attached faces. Meaning that, at this level, any remaining straight concave edge is considered for cut.

For the 7th level, all the remaining types of concave edges, including curved edges attached to any type of face, are considered. This stage has the highest possibility of generating unwanted features especially for machining or manufacturing purposes; the reason for this is, in general, these edges have complicated geometries attached to them and therefore are more likely to generate complicated features.

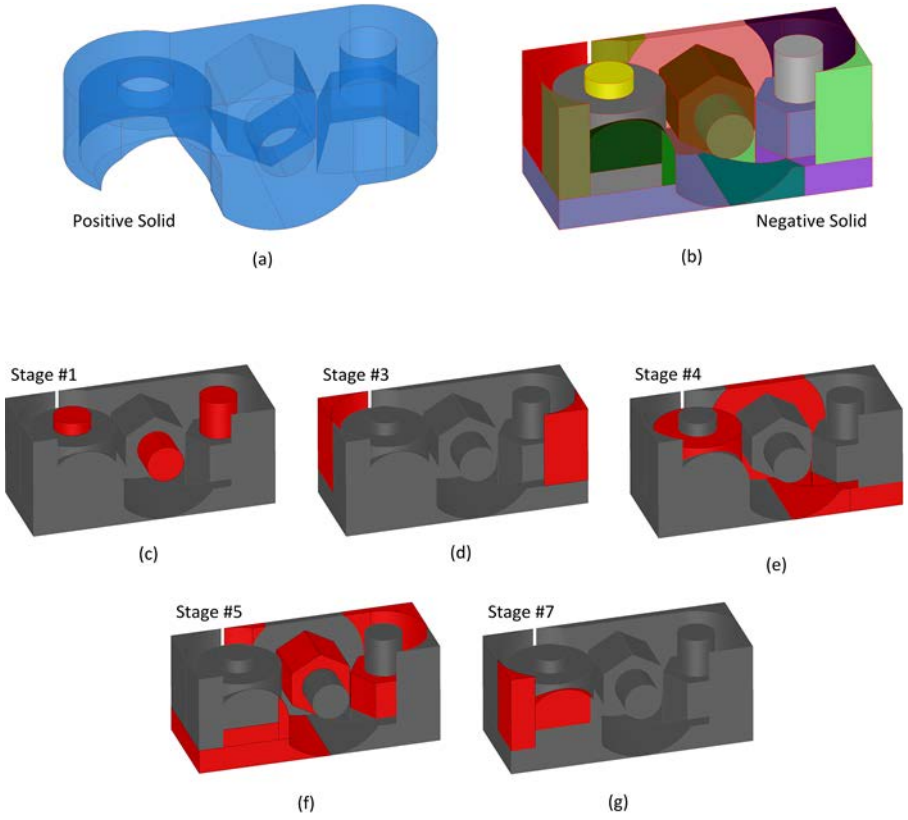


Figure 7. Positive Solid (a), Negative Solid (b) and Different Stages for Generating Compound-Negative Solid (c - g)

3.1.8 Results of Geometric Representation and Reasoning

The capabilities of the proposed algorithm have been tested in several examples resembling real automotive parts. The method is implemented in C++ with Open CASCADE API on a computer with an Intel 3.4 GHz processor and 16.0 GB of memory. The first example is shown in Figure 7. The positive solid shown in Figure 7 (a) is similar to the positive in Figure 3 (a) but upside-down. As shown here five out of seven stages in concave-edge-selection flowchart are involved in cutting the solid in the convex pieces shown in Figure 7 (b). Figure 7 (c) shows the result of stage 1 in decomposition flowchart. In this stage, all cylindrical features are captured and isolated using full 360° circular cutting edges. There are 4 cylindrical shapes existing in the

original solid but only 3 of them are captured at this step by the algorithm. This is because the largest cylindrical volume has interactions with other features in the geometry and hence the cutting edge related to this geometry is not a full circle; therefore, it is not captured in this stage. Stage 2 is related to half-circular concave edges, and since there are no edges of this type in the solid model, this step is not involved in the decomposition. Figure 7 (d) is related to type 3 concave edges. Two volumes that correspond to two fillets in positive solid are isolated here. It is important to note that although these volumes have concavities, they are considered as convex shapes here. As mentioned before, this is because the criterion for decomposing 3D solids in this work is the availability of only sharp concave edges in the original volume. However, one can argue that curved concavities, such as cylindrical surface in the highlighted volumes can also be used to generate further cutting directions but, in fact, they will generate infinitely many small sub-convex pieces due to the nature of their geometry. This is not suitable for process planning applications and hence is not considered.

In stage 4 shown in Figure 7 (e), cutting directions that are related to curved edges with at least one planar face attached to them are involved in the decomposition. This stage results in 5 convex shapes that are marked with different colors in the Figure. Figure 7 (f) shows the result of decomposition in stage 5 where straight edges that are attached to only planar faces are considered for cut. The result is 7 distinct features that are captured here and are shown in different colors. After this step, since there are no edges of type 6 in the geometry, stage 6 is skipped. The last two partitions are shown in Figure 7 (g) and are captured in stage 7 with the remaining cutting curved edges. Since this is the last step, it generates the final decomposed solid as shown in Figure 7 (b). It is important to note that in general there are multiple options at each stage for choosing the desired volume (cutting side L or R), where each option generates different solutions. According to our exploration of a number of these solutions, the presented decomposed solid is the most suitable for our manufacturing reasoning framework. Detail of the subsequent reasoning is presented in Fu et al. [2]. Therefore, the generated compound-negative (shown in Figure 7 (b)) is considered as one among approximately 1,000,000 (2^{20}) possible options for this model since there are 20 concave edges in the original solid and 2 cutting directions per edge.

As mentioned above, the solution that is presented here as a compound solid is more suitable for machining applications. Based on the developed method, the decomposition algorithm only generates one feasible solution from a pool of options available in the decomposition tree. In order to tell the difference between a good and a bad result, let us explore an erroneous case where concave edges are chosen in a different order than what we have presented here. For simplicity let us only look at a case where the orders of type 4 and 5 concave edges are altered in the flowchart of Figure 6. In this case straight edges with any type of face attached to them have priority for cut compared to curved edges that are attached to at least one planar face. The result of this change is shown in Figure 8. The highlighted partition wraps around 3 faces of the bounding-box. This is spindly and not compact, and is clearly a non-machinable volume. Hence it is not suitable for manufacturing planning purposes. Concave edge classifications and heuristics designed in this work ensure that all the generated partitions consist of simple convex-volumes.

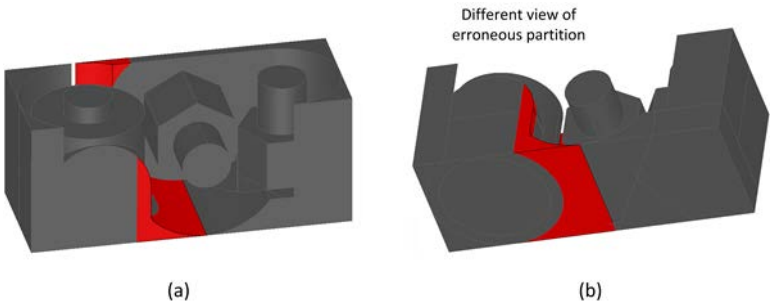


Figure 8. Decomposed Solid with Erroneous Partition

The example in Figure 9 shows a more complex solid model because there are a lot of interacting features that exist in the shape. As illustrated, the original solid is shown in Figure 9 (a). By taking a subtraction between bounding-box and original solid and applying the convex decomposition algorithm we get the decomposed solid in Figure 9 (b). The total numbers of convex regions are 28 in this case. Note that there are several cylindrical features that formed complex interactions with neighboring sub-volumes and are successfully isolated and decomposed from the solid model. The example in Figure 10 shows a case where rounded concave corners in the original solid Figure 10 (a) have interactions with chamfer-like features and are decomposed into different sub-volumes in Figure 10 (b). Note that the algorithm also captures the countersinks on the holes. The final example is shown in Figure 11where there are 9 partitions in the negative solid. The big rounded shape in the original solid generates two complicated geometries in the negative solid, but they are successfully decomposed into two simpler volumes with cutting edges that are derived from holes in original solid.

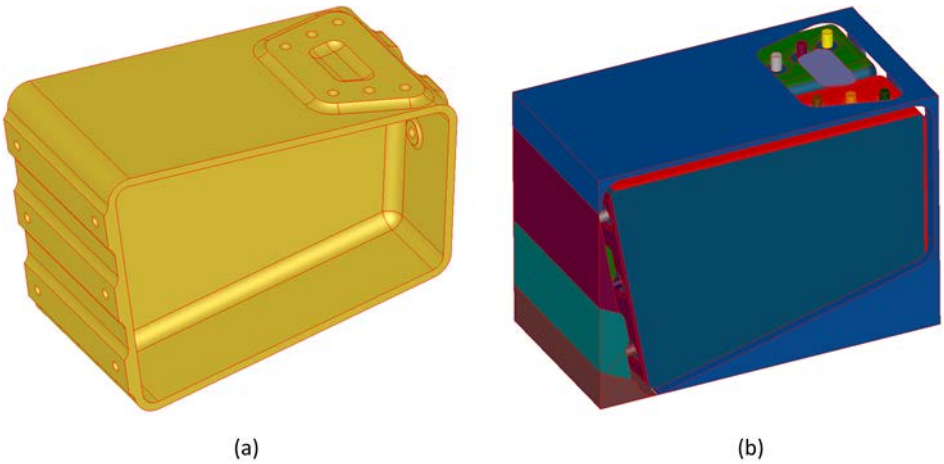


Figure 9. Positive Solid (a) Decomposed Negative Solid (b) with 28 Partitions

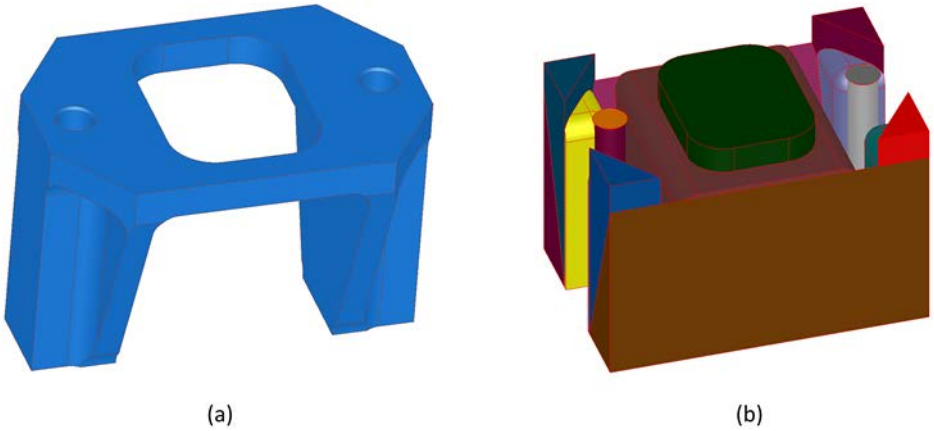


Figure 10. Positive Solid (a) Decomposed Negative Solid (b) with 15 Partitions

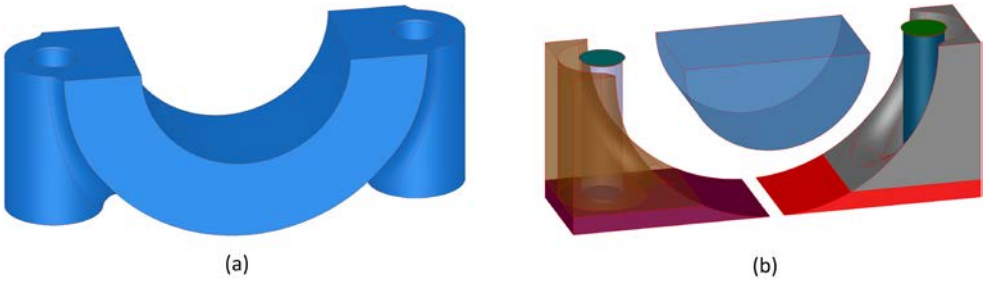


Figure 11. Positive Solid (a) Decomposed Negative Solid (b) with 9 Partitions

3.2 Graph Grammar Rule Library for Foundry Machine Representation

Our approach is to use the grammar representation to reason about the manufacturability of a given part under certain foundry capabilities. To begin, all available manufacturing processes within a foundry are translated into grammar rules. The rules are then organized to reason about the seed graph in order to determine its machining details. A search tree is drawn to describe how they work on the seed graph. Steps in the tree represent alternative manufacturing operations for different sub-volumes. These operations are determined through the rules which detect a series of graph elements and relate them to a particular manufacturing process. Each operation consists of the tool entry face, the tool type choice, the machine choice, and the needed fixture to machine one sub-volume. As the tree grows, more and more sub-volumes get manufactured. When the tree propagates to its bottom, there are no more sub-volumes of the given part to be machined, and a complete search space that includes all alternative manufacturing plans for the given part is derived. In addition, by translating foundry capabilities into graph grammar rules, a precise conclusion of non-manufacturability of a part can be made if the rules fail to find a manufacturing plan for a part. It signals that the manufacturing process is beyond the foundry capability and this part needs to be redesigned.

This section describes the grammar representation and shows how the method functions on a few test parts. First, a description of how the STEP file is converted into the seed graph for use in the

Grammar Representation is provided. STEP (Standard for the Exchange of Product model data) is one of the most common CAD data exchange formats in modern systems. Then we present the seed format and the rules that reason with the graph. After that, a tree search algorithm is described which used the representation.

3.2.1 Seed Lexicon

After the removal volume of a given solid model is decomposed, the compound solid comprised of different sub-volumes has to be translated into a seed graph such that the grammar representation can work on it. Rather than using existing graph techniques to represent a solid model, a new lexicon is proposed. Figure 12 gives an example showing how a solid model is described as a label-rich graph.

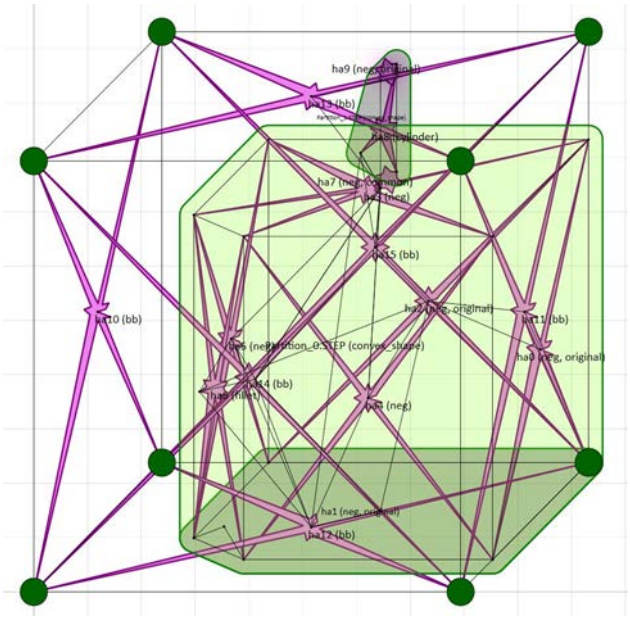


Figure 12. A Seed Graph in Representation

This example is a simple shape with a pocket in front and a through hole in the back. In this graph, geometric elements are described by nodes, arcs, and hyperarcs. Nodes are used to represent vertices and faces. Arcs are used to represent edges in the shape as well as to indicate relative positioning information (parallel, perpendicular, etc.) between any two faces. A hyperarc is a special arc. While arcs can only connect two nodes, a hyperarc can connect as many nodes as needed. It is always used to connect all vertices belonging to a face to their face node. It is also used to encompass a sub-volume by connecting all of the nodes of a sub-volume together. For example, in Figure 12 the hole and the bottom cuboid are separated by two green hyperarcs. By using nodes, arcs, and hyperarcs in the seed graph, all geometric information about vertices, edges and faces for a solid model is stored. In this way, face nodes are used in the seed and rules to refer to general machining features, like holes, pockets and slots. Moreover, edges and vertices

provide more details about shapes and geometries, which are necessary for the rules to describe manufacturing operations more precisely.

It is also important to note the variety of labels in the graph, which are used to store topological, rather than parametric, information. A face node may have label “bb”, which indicates that this face is a bounding box face. If a face node represents a face belonging to the removal volume, it will have the label “neg”. For example, in Figure 12, the face node n1 (not explicitly shown, but overlaps with its face hyperarc ha1), which represents the bottom face of the cuboid, has label “neg”, while face node, n12, has label “bb”. A hyperarc may have label “original”, which means the face this hyperarc connects to is an accessible face to the tool and it is a candidate face for the tool entry face selection. Examples include the hyperarc ha1 and ha9. Besides, the face adjacency property, either convexity or concavity, between any two adjacent faces is stored in the label of their common edge. With these labels, the rules can do a much more precise reasoning about the graph elements they capture and the search can be more successful. The detailed explanation of the rules based upon the label-rich seed graph is given in 3.2.2.

3.2.2 Rule Development

With an understanding of how a compound solid model comprised of different sub-volumes for a given part is represented in the seed, it becomes easier to tell how the rules are developed. In this part, eight sets of graph grammar rules have been devised to simulate a virtual machining process, removing material of the compound solid step by step. This process will end if the volume of the compound solid becomes zero. These rule sets are arranged in a specific sequence such that they collectively perform the required reasoning as a whole.

The first rule set (rule set 0), the pre-processing rule set, aims to recognize typical sub-volumes (countersink, round edge, etc.) and non-traditional machining operations (bending, etc.) which are tagged for later use. These sub-volumes are usually machined in a finishing process using specific tools. By recognizing and isolating these special cases at the first stage, more realistic manufacturing plans which separate roughing and finishing processes can be generated. Unlike other machining operations, bending operations do not remove material. They simply change the shape of the seed graph. However, this change does affect the generation of a correct bounding box for a given part. In this case, the rules in this rule set operate in cooperation with the Convex Decomposition to implement these non-material-removal operations on the part before a correct bounding box is generated.

The second and third rule sets (rule set 1 and 2) are used to identify a feasible tool entry face on a sub-volume that is to be machined. Firstly, in rule set 1, a single rule is designed to capture a face which is accessible by the tool. If such a face is found, it will be labeled with “machining_start”. If no faces are found, the process terminates – there are no sections left to machine. In rule set 2, there are two rules, representing two special cases, where a tool entry face selected from rule set 1 must be rechecked. For example, an accessible face is not allowed to be chosen as a tool entry face if a sub-volume could not be fully removed from this face. An infeasible tool entry face is described in Figure 13 using the same solid model as discussed in Figure 12. If the hole is first removed, the internal circular face of the hole that is shared by the front pocket becomes accessible to the tool. But this face is not a good choice of the tool entry face since from this face

the tool cannot access the whole material of the pocket. Despite being a valid face to begin machining in rule set 1, the choice is invalidated in rule set 2.

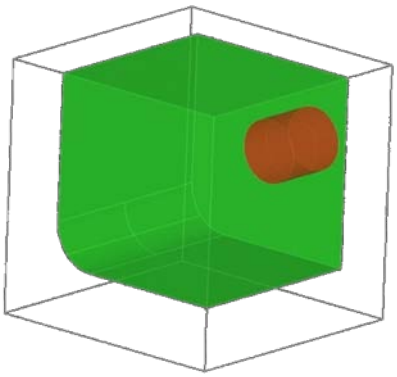
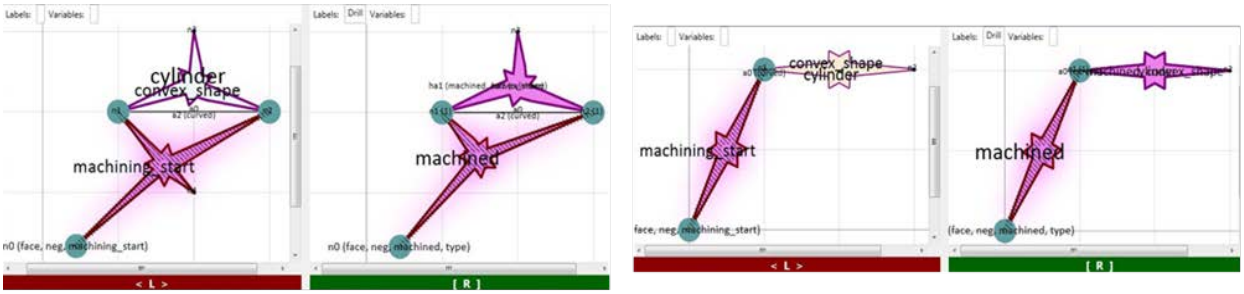


Figure 13. Example of Infeasible Tool Entry Face



(a) Drilling rule 1 for hole of type 1

(b) Drilling rule 2 for hole of type 2

Figure 14. Two Drilling Rules in GraphSynth [3]

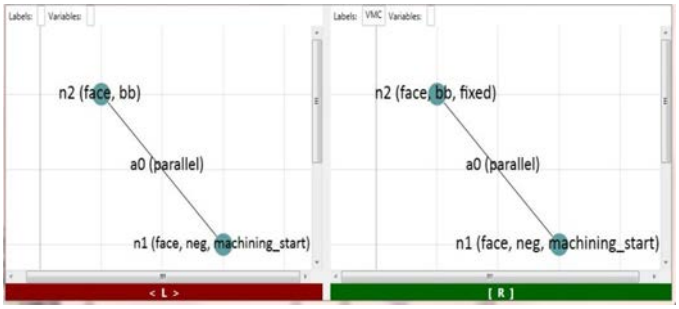


Figure 15. VMC Rule from Rule Set #5 in the Grammar Representation

After rule set 1 and 2, a feasible tool entry face is identified. Then, the representation goes to rule set 3, which is responsible for tool type selection. Rule set 3 is a cluster of available tooling operations, including Drilling, Milling (End Milling and Ball Milling), Sheet Cutting (Water Jet), and Countersinking. Each tooling operation corresponds to one or more rules in this rule set. These rules are specially designed based on how each operation is implemented. For example, in this rule set there are two drilling rules as shown in Figure 14.

The reason for creating two rules is that there are two different representations for holes in STEP files. The planar circular face of a hole can be represented with either two vertices and two semi-circular edges (type 1) or one vertex and one 360° circular edge (type 2). Figure 14 (a) recognizes the first hole type. The LHS of this rule finds a hole to be machined by capturing its cylindrical face (a hyperarc labeled with “cylinder”) and one of its planar faces, which is accessible by the tool and is depicted as another hyperarc labeled with “machining_start”. Additionally, since this hole is also a sub-volume to be machined, its sub-volume hyperarc (a hyperarc with label “convex_shape”) is also captured. If such a hole is found, a drilling operation is implemented on this sub-volume, which is described by a virtual transformation from LHS to RHS of the rule. After that, this sub-volume’s machining_start face and its sub-volume hyperarc become “machined”. Similarly, Figure 14 (b) is the drilling rule for the hole of type 2.

Similar algebraic reasoning is performed for the remaining tool types where complete Left Hand Sides have been developed to capture the intricacies of the geometric constraints.

After a tooling rule is selected and applied to a sub-volume, this sub-volume is marked as machined. Then the representation moves to rule set 4, which is a post-processing rule set. This set generally takes care of any remaining issues regarding label changes and graph modifications after each tooling operation. For example, a very often recognized rule in this rule set is used to delete any remaining faces which realistically should have been removed after a previous tooling operation.

Once a complete tooling operation is finished after rule set 4, the corresponding fixture method and machine type used to conduct this tooling operation are selected in rule set 5. Currently these two decisions are made in one single rule by considering which bounding box face to fix and the relative positioning relation between the fixed face and tool entry face of this sub-volume. If they are parallel, then a VMC, or Vertical Machining Center, is chosen. If they are perpendicular, a HMC, or Horizontal Machining Center, is recognized. The rule for the VMC is shown in Figure 15. The LHS of this rule captures a pair of parallel faces while one face is a bounding box face (the face node n2 labeled with “face” and “bb”) and the other face is a tool entry face (n1 labeled with “face”, “neg” and “machining_start”). If these two faces are found, then the bounding box face n2 will be “fixed” and the “VMC” will be chosen. Future endeavors for this rule set are focusing on extending machine types to multi-axis machines such that more general fixtures can be covered.

After fixture design and machine type selection, the following rule sets 6 and 7 are post-processing rule sets. Similar to rule set 4, they are doing some clean-up, like adding new original labels to those faces which become exposed to the tool after certain sub-volumes are removed.

After rule sets 6 and 7, one complete step in a manufacturing plan is defined to machine the current sub-volume. Then the representation will go back to rule set 1 to start another loop for another sub-volume. This representation loop is shown in Figure 16. If all the sub-volumes for a given part are machinable, the similar loop for each sub-volume will continue until all sub-volumes are machined. At that time, since there are no more tool entry faces for rule set 1 to choose, the looping process will terminate with a complete manufacturing plan for the part, which we refer to as a goal in the search tree.

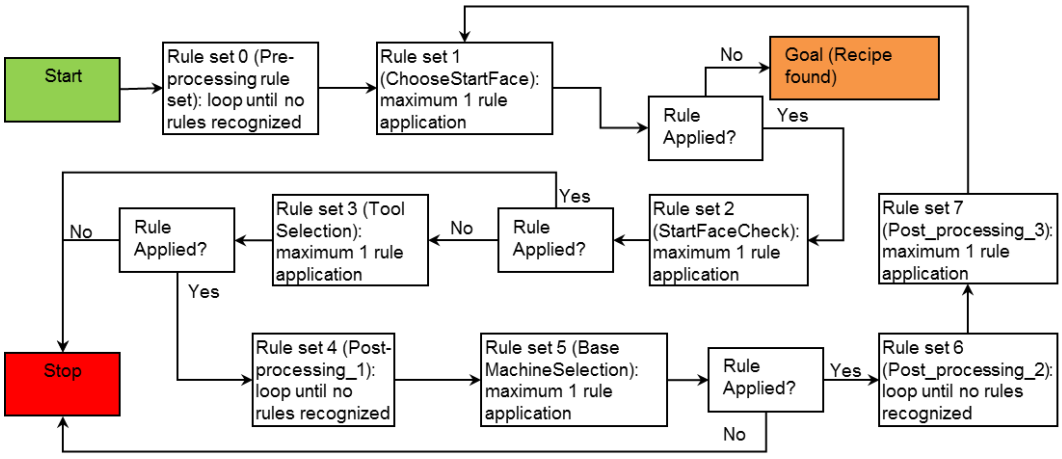


Figure 16. Flowchart of Rule Sets

However, as seen in Figure 16, rather than finding a complete machining recipe, the reasoning process can also end at different stopping points defined by different rule sets. Depending on the function each rule set performs, the representation loop ends at these stop points either when there is no rule applied in certain rule sets (rule sets 3 and 5), or when a particular termination rule is recognized (rule set 2). For the first scenario, for example, the looping process will stop if there is no tooling rule in the Tool Selection rule set that is recognized for a sub-volume. If this case happens, the user can gain an insight that a current sub-volume of a given part is actually not manufacturable with current available tooling operations described in the Tool Selection rule set. Since a foundry's capability is always mapped into different tooling rules, it is equivalent to conclude that the manufacturing of this part is beyond given foundry's capabilities. One can either redesign the part to make it easier to manufacture, or one can import more advanced machines and tools into the foundry to cover required tooling operations. For the second scenario, for example, the rules in rule set 2 define several infeasible cases of tool entry face selection. If any rule in rule set 2 is invoked, the tool entry face selected in rule set 1 becomes invalidated, and the search will terminate after this rule is applied.

Therefore, depending on the given part and the knowledge of the manufacturability analysis built in the rules, a complete looping process described in Figure 16 will either find a feasible manufacturing plan, or converge on no plan. One should be aware that this complete reasoning process only represents one branch in the search tree in Figure 17. The whole search process actually contains numerous branches where each branch is depicted as one complete looping

process in Figure 16. More detailed discussion about the heuristic search tree is given in the next section.

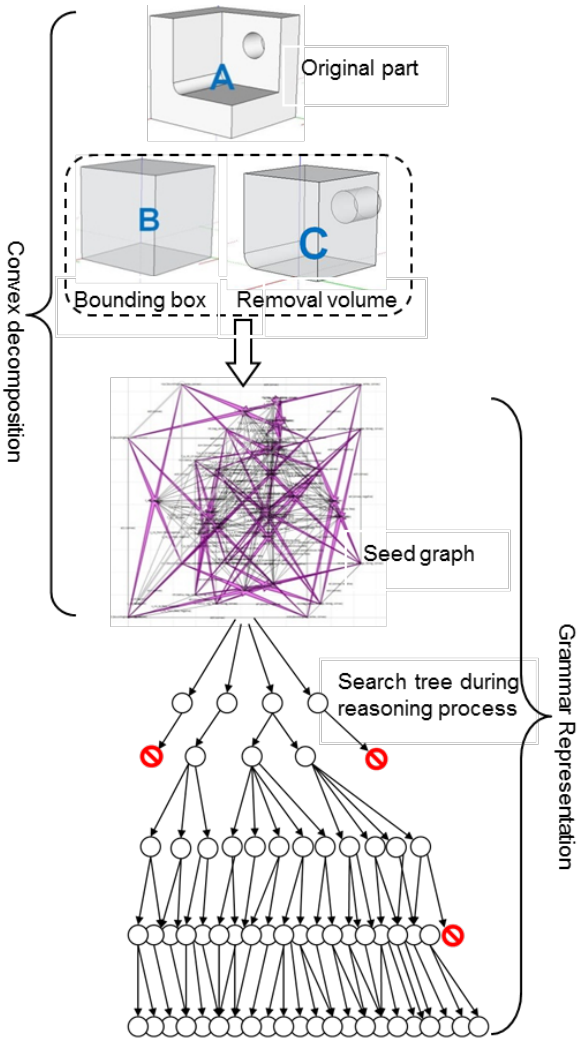


Figure 17. Flowchart of the Representation Scheme

3.3 Heuristic Search for Process Planning

Now that the seed graph and all necessary rules have been described, a search process using these rules is presented which seeks feasible and optimal manufacturing plans. A graph grammar based software tool called GraphSynth [4] previously developed by the research team is used as a platform where the seed graph can be loaded and a Recognize-Choose-Apply cycle is invoked to define the tree. During the search a candidate host graph is provided to a recognition procedure which checks all the rules within a single rule set to find valid rules that can successfully change the graph. This defines a list of options which are essentially different branches in a search tree (similar to Figure 17). Amongst these options, one is chosen. However, given the expanse of computer memory most tree-search algorithms choose all possible paths

thus defining a population of states. The Apply procedure executes the L-to-R graph transformation algebra to change the host state into a new graph.

In several of the rule sets described in 3.2.2 (0, 2, 4, 6, and 7), the options do not define meaningful alternative paths. The rules within these rule sets simply prepare, fix, or check qualities of the graph. Many of the resulting options are confluent which means that they result in graph changes which do not negate other options from being called. As such these rule sets do not define alternative decisions in the tree. However, the decision rule sets (1, 3, and 5) create decisions based on (1) the tool entry face where machining is initiated, (3) the choice of tool to use, and (5) the type of machine and fixture orientation to use.

We surveyed various AI search algorithms before deciding to use GraphSynth. Our survey included various planners, constraint solvers, as well as logic solvers such as Prolog. Choosing GraphSynth, we implemented a depth-first search of the tree of valid manufacturing plans was defined which used the rules to define the successors at each state in the tree. When no successors are identified through the tree-search, then the resulting plan is either complete or a dead end. Dead end candidates are discarded and completed plans are stored and evaluated. In order to perform the evaluation, the recipe of rules that are called for each candidate is converted to a recipe of machine operations. This new recipe is both easier for humans to understand and provides the proper inputs for the evaluation routines.

The search algorithm uses various pruning techniques. First, we are not concerned with inferior plans, because we wish to display only the best process plans to the designer. To this end we implemented functions to support detection of Pareto-dominated process plans. Such plans, when detected, are simply pruned from the search which reduces the number of plans the user has to sift through as well as improves the performance of our solver. Second, we detect duplicate sub-problems and cache intermediary results. The search algorithm has access to information about which sub-volumes must be machined, and in several cases, many of these sub-volumes are relatively independent with one another. A naïve algorithm would explore all possible permutations of processing these sub-volumes. Instead, we implemented a powerful duplicate detection technique to reduce the size of these permutations. For a given subset of sub-volumes that have been machined, our algorithm remembers the particular subset, so that if we encounter the same subset in a future search sub-tree, we can simply skip over it. This technique resulted in over 30 and 40-fold speedups in several of our test cases, including test parts provided by Rolls Royce, making representative challenge parts solvable within under a minute.

We also implemented two methods of obtaining a more diverse sampling of the various process plans in the search space. Especially in cases where we are unable to completely enumerate all process plans, we at least present to the user a set of plans that adequately cover the search space. As an improvement to our best-first search algorithm, we introduce randomization when generating children nodes during the search to ensure a diverse sample. We have also implemented beam search, another search algorithm that complements our best-first search algorithm, which also features diverse sampling of the search space. Due to this sampling, the logic behind the Pareto domination algorithms was much more complicated. In particular, we had to compute the convex hull over a set of points, and used that for testing Pareto domination. Because of the many additional calculations, we ran into the problem of floating-point rounding errors, and therefore had to extend our code to explicitly handle these precision issues.

We also have extended the functionality of our solver to allow finer control over the search such as trading off the model fidelity with the branching factor and overall computational requirements with respect to our model for machining time and cost due to tool wear. Other additional supporting functions such as cutting off the search at a particular depth and being able to turn on and off various techniques simply via a command-line switch facilitate more rapid software development of our search algorithms.

Finally, our search algorithm captures the hierarchical way that we reason about machining various sub-volumes, which resulted in significantly faster performance compared to a solver without the technique, allowing us to handle larger and complex parts (such as the parts from the most recent META benchmarks) in a shorter amount of time. This addresses the problem where a search algorithm might reason over all potential subsets of machining sub-volumes in the desired part in conjunction with reasoning over how to machine each individual sub-volume. Considering both problems simultaneously multiplies the difficulty of one with the other. By first enumerating all possible ways of machining each sub-volume individually, we can provide all such options to a dedicated and optimized scheduler which will pick and reorder such options very quickly. This decoupling of the two problems significantly sped up our solver.

We have pulled in an open source planner and scheduler, called Fast Downward [5], and have code that translates a sub-problem in our domain into PDDL (Planning Domain Definition Language), the general planning language. This allows our search algorithm to delegate the sub-problem of choosing which machining plan to use for each feature as well as the scheduling of such plans to the dedicated solver. Our encoding includes accessibility constraints, which captures which faces of which subvolumes are now accessible due to machining operations on other subvolumes. It also captures constraints relating to the cost of switching tools, switching machines, and fixturing on a different face. This enabled the planner to optimize against time and cost.

3.3.1 Results and Discussion of Process Planning

The current grammar representation was tested on over 20 solid models, ranging from self-designed simple geometries to complex suspension components provided by Rolls-Royce. Two examples are provided, including one easy part as shown in Figure 17, and one real part, which is a chassis part for small vehicle. In the simple example, the generated recipes will be verified. For the main chassis, selected recipes will be explained.

For the simple part, one may easily come up with a manufacturing plan. One such simple plan may be to first remove the bottom cuboid from the bottom face ha1 (as indicated in Figure 12), then drill the top hole from face ha7. In this way, only one fixture is needed. However, in a complete search tree created by automatically employing the rules on this seed graph, this plan is only one branch. All other branches representing different alternative plans are simultaneously generated. One sample plan is shown in Figure 18.

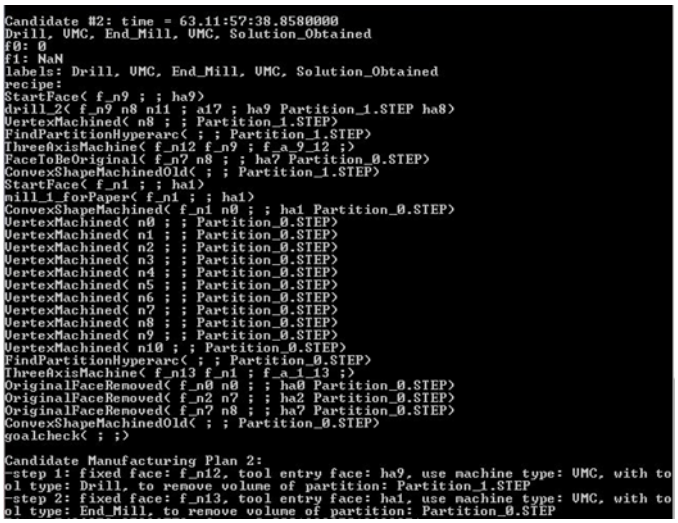


Figure 18. Sample Manufacturing Plan for the Solid Model in Figure 12

In this manufacturing plan, the upper portion is a step-by-step description of how the rules were implemented. An executable machining plan was provided at the end. Based upon this recipe, two steps are needed to make the part. First step is to fix the bottom face and drill the top hole from its top surface. In this case, a VMC is needed to finish this operation. Then the part is flipped over and the top face is fixed. The remaining cuboid is end milled from its bottom surface. Of the 96 valid plans that are found (as discussed next), there are several inefficient plans like this one, and several of the more likely single-fixture approach.

Figure 19 is a summary report after all possible manufacturing plans have been found. Within 2 seconds 96 solutions were generated for this simple part. The completeness is reflected by the total nodes in the search tree while the efficiency is reflected by the search rate, i.e. Nodes/Sec. One may doubt the large size of solutions. However, this number can actually be verified by a pure mathematical derivation.

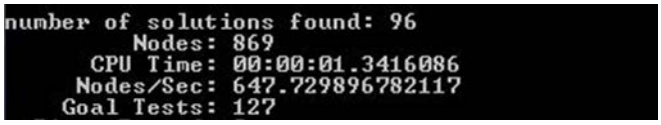


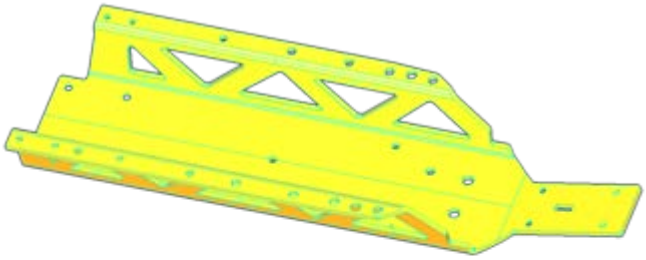
Figure 19. Summary Report of Search Space from part in Figure 18

If starting from ha1, only the End Milling tool is capable of removing the bottom cuboid. Despite the three bounding box faces that are coplanar with the outer surface of this cuboid where machining will happen, there are three other faces available for fixturing. After removing the cuboid, the hole can be either drilled or end milled from either its top face or its inner planar face. For the first case, there are five kinds of fixture, and for the latter one, there are six. So in total, $1\times3\times(2\times5+2\times6)=66$ solutions are found. However, the upper face, ha9 can also be chosen

as the first entry face. Both End Milling tool and drill bit can be used to remove the hole. Five bounding box faces are available for fixture. After the hole, the cuboid can only be End Milled from ha1 with three different fixtures. So another 2×5×1×3=30 solutions are found. Therefore, in total 96 candidate plans are derived, exactly the same number of solutions found by the software.

This example shows the validity of the grammar representation scheme. The grammar rules are good at creating a complete search space, which covers all possible and realistic candidate manufacturing plans for a given part, while the geometric reasoning about feature interactions is properly conducted by a specially designed search sequence for the rule sets.

Another example given is a main chassis part for a small autonomous vehicle (Figure 20). This part is special due to its tilted head, one rectangular through pocket and countersinked holes. Apparently, it is manufactured from sheet metal with final bending operations. In addition, non-traditional machining operations are needed to cut out sharp corners for the pocket. For simplicity, some duplicate holes and triangular pockets were removed. However, even the simplified part (Figure 20 (b)) has 12 sub-volumes, which represent at least 12 steps to machine this part. Due to the complexity, the seed graph converted from the simplified model is not shown.



(a) Original main chassis part

(b) Simplified main chassis part

Figure 20. Main Chassis Part

For this part, the preferred manufacturing plan is to first cut out the sheet metal with exactly the same shape, then either machine the holes or cut off the pockets, after that machine all countersinks continuously, and finally do several bending operations. However, since the software first looks at the final part then does a backward reasoning of how it was machined; one may expect that bending operations, instead of being the last few operations, are actually replaced by unbending operations at the very beginning of a generated manufacturing plan. For the same reason, the seed graph used in this case was not converted from the simplified part. As mentioned in 3.2.1, rather than removing material, bending operations significantly change the shape of a given part. As a result, the right bounding box and negative part could not be

generated from that simplified part (Figure 21 (b)). To solve this problem, a special heuristic is designed in the Convex Decomposition such that once a bending operation is detected for a given part, all needed pre-unbending operations are executed in the convex decomposition first, and then the unbent part is used to generate the seed graph. In this case, the seed graph was converted from a sheet metal part which has already been unbent from the simplified main chassis part. A sample recipe is shown in Figure 21.

Figure 21. Sample Manufacturing Plan for Chassis Part

From the recipe one can see that actually five unbending operations are needed before machining process starts. After that, the Water Jet is used to cut most of the sub-volumes. It is very important to note that counter sink operation always goes after drilling operation to satisfy the manufacturing constraint that there is no hole before the hole is drilled.

The complete search space of all possible manufacturing plans for this part expands exponentially as the total number of sub-volumes increases. For the simplified part with 12 sub-volumes (Figure 20 (b)), an estimation of the size of the search space is conducted from the fact that there are at least one tool entry face, two fixtures and machine types and three tooling operations available, which in total represents 6 different options, for each sub-volume to choose. Moreover, these 12 sub-volumes can be manufactured in a random order. Therefore, in total at least $6^{12} \times 12! \gg 20$ trillion solutions are included in the search space. This underscores the need for an approach to eliminate many of these solutions to focus on more beneficial solutions. Ongoing evaluation research for the optimization of the search space generated in the representation is discussed next and the details are given by Blarigan [6].

3.4 Manufacturing Metrics

The metrics of interest for the machining of a partition are the time and cost associated with the process. By generating these metrics for each operation, we are able to determine an overall time and cost estimate for the production of entire parts. To perform the calculation, we simulate the machining process, including tool selection, path planning, and machining parameters. The non-machining production steps (part inspection, fixturing, etc.) are also analyzed to capture all processes that a part must go through in the manufacturing process. Once all times are known, a

cost can be assigned to the operation from the time values and costing information of the foundry.

3.4.1 Tool Selection and Path Planning

If the representation step of assigning a machine start face has occurred, the evaluation method may perform tool selection and tool path planning on that face, in order to simulate 2.5D milling. A few requirements and assumptions are critical to the success of the method, these are: machine start face will always be convex, and any square internal corner is not a real boundary. To address the second case, it is important to note that convex decomposition will create a closed loop face for each partition. This partition may, however, have edges open to the air or other partitions that require machining. This will create partitions with sharp internal corners where there should be an open boundary. Since representation will be able to catch any real sharp internal corner as a non-machine-able situation, we assume in evaluation that any sharp internal corner is an open boundary, and neglect the fact that the tool path leaves material in the corner.

3.4.2 Tool Selection

The automatic tool selection method presented here operates with offset curves. The shape of the curves is never used, only the length and distance offset from the boundary.

The pocket at the top of Figure 22 is a feasible boundary curve, with offset curves generated until no further offsets are possible at the increment chosen. What is stored in the partition is the array shown at the bottom of the Figure. This array is named ShapeData, and will be referred to as such here. ShapeData[0] always contains the smallest radius in the boundary curve. If there is no radius in the boundary, this cell will contain 0.0. ShapeData[1] contains the chosen increment for consecutive offsets, in this case, 0.5 mm. ShapeData[2] is the length or perimeter of the boundary curve. From ShapeData[3] until the last entry, the numbers represent the length of the offset curves. Each position represents another increment from the previous number. For Example, ShapeData [3] = 0.5 mm from boundary, ShapeData[4] = 1.0mm from boundary, etc.

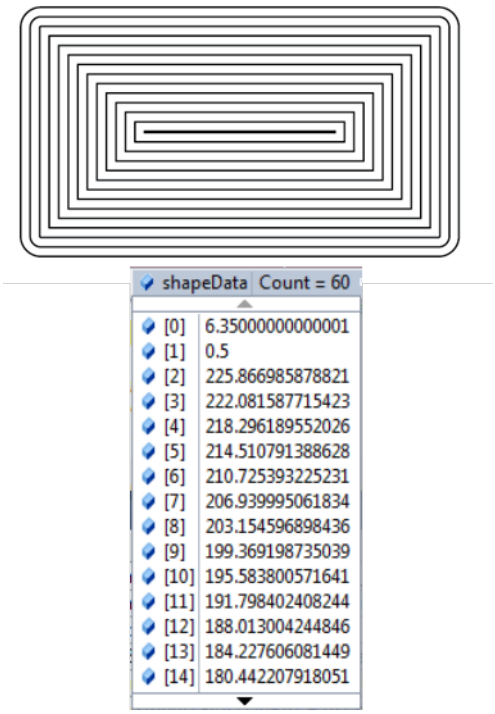


Figure 22. Conversion of Offset Curves to ShapeData

The specific format of the information stored reveals information about the original shape. The first number in the array is always the radius of the smallest curvature in the path. This allows for rapid assessment of what the smallest tool required to machine the area is, since the tool must have a radius smaller than the smallest radius of curvature on the part in order to not leave any residuals.

The desired tool size is chosen to be slightly greater than half the narrowest internal dimension. This selection closely matches the typical tool selection done automatically with CAM packages or by machinists. The diameter of the tool is then calculated using Equation (1.

$$R_{tool} := N_{\#\ of\ offsets} * R_{offset} + R_{offset} \quad (1)$$

The geometric equivalent of Equation ((1) is shown in Figure 23.

Equation (1 is used even for non-square or rounded shapes, although the area accessibility will change for each of these scenarios (notice the large amount of inaccessible area to a tool of this size for the triangular geometry shown in Figure 23.)

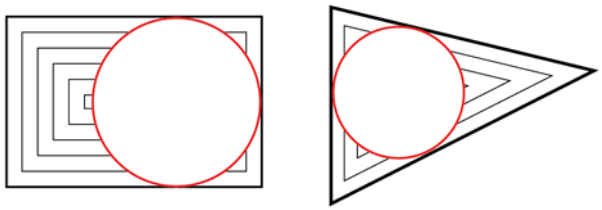


Figure 23. Maximum Tool Diameter for Accessibility

The extra R_offset term in Equation (2 is to ensure that the diameter of the tool chosen exceeds half the size of the feature, so that if it traverses the boundary curve, no island of material is left in the middle. It is appropriate because the final offset will not reach the center, due to the finite increment value. The distance from the final offset to the middle could be anywhere between 0 and 1 increment, so adding an increment to the diameter ensures that the middle is reached. Should the machinist prefer to use smaller tools, this is easily adaptable to selecting a tool that is any size less than that given by Equation (2.

$$D_{tool} := N_{\#\ of\ offsets} * R_{offset} + R_{offset} \quad (2)$$

The geometric equivalent of Equation (2 is shown in Figure 24.

In some situations, this large tool is all that is required to be used. This case exists when the smallest radius in the boundary (given in ShapeData[0]) is smaller than the tool diameter, given by Equation (2, or is zero. For the two shapes given in Figure 24, this tool would be the only tool required to machine the partition, since there are not radii in the boundary. This means that the tool has 100% area accessibility, since sharp corners represent open boundaries, as discussed Section 3.4.1. If, however, the shape has a corner with a small radius, the part has area that is inaccessible by this tool, which necessitates the use of a secondary, smaller tool. The size of this smaller tool is easy to select, since the first entry in the offset curve data gives the radius of the smallest radius corner.

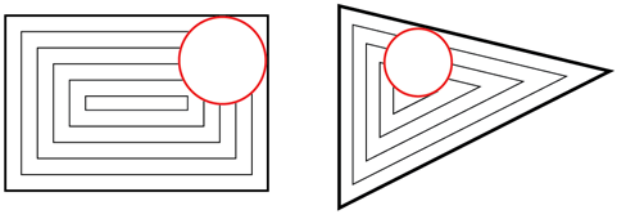


Figure 24. Chosen Tool Diameter for Machining

Although the size of each of the tools is easy to compute, we need a check to determine if two tools will be required for the partition. This check is simply a check between the diameter of the large tool and the diameter of the small tool, and is computed as follows:

$$R_{bigtool} - R_{fillet} \geq X\ (mm) \qquad (3)$$

If Equation (3 is true, two tools will be selected. The parameter (X) is a user-defined value. Currently this is set at 5 mm. Having this as a fixed value is a simplification. In future work an optimization routine would be implemented to determine when there is a time savings with two tools over one.

In addition to calculating the ideal tool size(s) for machining the feature it will also find actual tools available to the machinist. Using a database of available tooling, the software will load tool data for a tool that most closely matches the parameters calculated here. For the large tool we are looking for a tool that has the closest diameter to that given by Equation (2, with the requirement that the actual tool diameter be larger than Equation (2. For the small tool, if required, we are looking for a tool that most closely matches ShapeData[0], the smallest radius in the partition, with the requirement that the actual radius be smaller than the desired radius. Therefore we scan the database and load which tool satisfy the criterion:

$$\begin{aligned} &min\{\ BigD_{actual} - BigD_{calculated}\} \\ &\text{s.t. } BigD_{actual} - BigD_{calculated} \geq 0 \end{aligned} \qquad (4)$$

$$\begin{aligned} &\min\{|\mathrm{SmallD}_{\mathrm{actual}} - \mathrm{SMallD}_{\mathrm{calculated}}|\} \\ &\text{s.t. } \mathrm{SmallD}_{\mathrm{actual}} - \mathrm{SMallD}_{\mathrm{calculated}} \leq 0 \end{aligned} \qquad (5)$$

One common problem is when the calculated size for the large tool is larger than any tool in the database. The database is limited (currently) to 1 inch diameter, and it is common for partitions to be larger than this. In this case the largest tool available will be chosen.

Although we are currently selecting a tool based only on diameter, future work will include tool material, number of flutes, coatings, etc. into the selection process.

3.4.3 Tool Path Planning – Big Tool

Once tools have been chosen for a partition, in order to estimate the time to machine the part, we need to determine the amount of time required for the tool to remove the partition. This estimate will be based on the linear feed rate of the machine, calculated in the machining model in Section 3.4.7, and the path length of the tool. Once again, the offset curve data stored in ShapeData will allow us to estimate the path length required of the tool(s). Assuming we have selected a large tool using Equation $D_{tool} := N_{\#\ of\ offsets} * R_{offset} + R_{offset}$ (2, we know that the tool is slightly more than half the internal dimension of the partition. This is significant because it means with one sweep of the tool tangent to the boundary curve, the tool will remove

all material accessible to it. The curve that is offset from the boundary an amount equal to the radius of the tool is the centerline along which the tool would travel to achieve this. To prevent using a fully embedded tool, the final offset curve is also chosen, to allow the tool to cut a small pocket of material along the short inner curve, before moving to the outer curve. This curve is added regardless of the tool size chosen. If a smaller tool is used, the offset curves are selected such that their offset from the boundary is multiples of the tool radius. An implementation of this offset curve selection method and comparison to a FeatureCAM (commercial Computer-aided manufacturing (CAM) software for milling machines [7]) cutting path for the same shape can be seen in Figure 25. Figure 25(a) shows an internal pocket with offset curves stored in ShapeData. Using a tool that is 25% max tool size for accessibility, offset curves selected for tool travel have been highlighted in red in Figure 25 (b). In Figure 25 (c), a FeatureCAM simulation for the same model is shown, where the productive portion of the tool path can be seen in blue. It is clear that the tool paths in Figure 25 (b) and Figure 25 (c) are nearly identical.

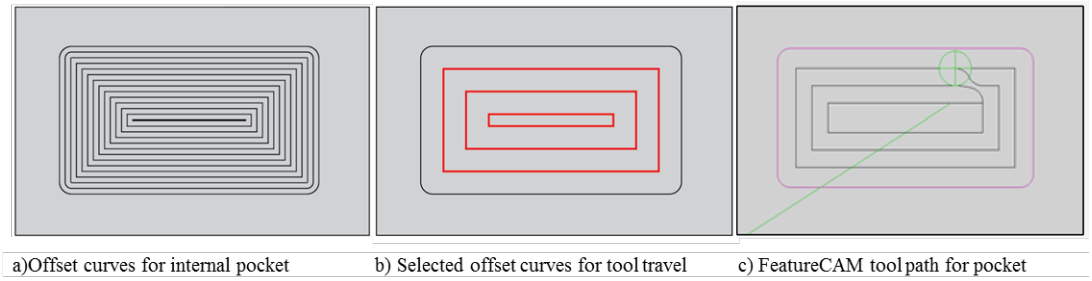


Figure 25. Comparison of Evaluation Method and Commercial CAM Tool Path

One difference that can be observed between our tool path curves and the tool path created by FeatureCAM is the tool motion between curves. We need to account for tool motion to jump from an inner curve to an outer one. As the tool path in Figure 25 (c) shows, this travel is more complicated than the distance between the curves. While FeatureCAM uses a variety of curves to move between cutting curves, we assume that the jump is done at a fixed angle from the curve. After some trial and error, an angle of 20 degrees has been found to most closely replicate the distance travelled by FeatureCAM, although this is an easily adjustable parameter in the model.

3.4.4 Tool Path Planning – Small Tool

The primary complication of using a secondary tool is being able to simulate that we do not need to recut everything the primary tool was able to access. Since we have no geometric information about the shape or even the offset curves used as a tool path for the large tool, the approach we take is to estimate the planar area accessed by the larger tool, and take the difference between this area and the face area as the amount of material that the small tool is required to remove. This requires the determination of the area removed by the large tool.

3.4.5 Area Removed by Big Tool

Given the offsets selected to be part of the tool path for the big tool, we can determine the area accessed by this tool. To do this, we must be able to estimate the area between any two offsets, given the increment that separates them. If the separation between them is R, and the length of the outer curve is L, the area between these two curves is given as:

$$Area\ between\ curves := L * R - X * R^2 \quad (6)$$

where X is a coefficient that depends on the number of sides of the curve, amount of boundary that is curved, and overall shape of curve (variance in internal angles). Since none of these values are known, the value of X must be estimated. A study was performed on the possible range of values of X, the results of which report X can range from pi to 11.7, with the typical value for our simulation around 4. The details of this analysis will not be presented here, and 4 is used as the coefficient, X, for all calculations. The second term in Equation $Area\ between\ curves := L * R - X * R^2$ (6 shall be called the corner adjustment term, and it is the only source of error in this estimation. Note that the error depends only on r, so as L becomes large with respect to r, the relative error becomes very small.

To determine the area, we start at the innermost curve and move outwards, so we will add the corner adjustment term rather than subtract it. We will also make an adjustment for the fact than on the final pass, the exact area of removal is known, since the radius of the tool is known.

$$Area\ removed\ by\ large\ tool :=$$
$$\sum L_{toolpath} * R_{tool} + 4 * (N_{offsets} - 1) * R^2 + \pi * \ R^2 \quad (7)$$

Where L is the length of each offset selected to be part of the toolpath, and R is the radius of the tool. The difference in Equation $Area\ removed\ by\ large\ tool :=$

$\sum L_{toolpath} * R_{tool} + 4 * (N_{offsets} - 1) * R^2 + \pi * \ R^2$ (7 and the area of the face (known) will give an estimate of the area of the face inaccessible to the large tool, which the small tool will have to remove. Translating this value to a path length requires another trick, using the step-over value for the tool. Given the diameter of the small tool, and the area the tool must remove, dividing these values gives the length the tool would have to travel to cover the area. This does not account for unproductive tool travel, for example, when the tool must move between unconnected sections of residual material, which can be a significant portion of tool travel for a cleanup tool. The assumption is made that since these shapes are convex, the area left to be machined is in the corners of the shape, so the length of the boundary curve is added to the tool path for the small tool, to account for the fact that the small tool must travel to all of the corners to machine them. Therefore the small tool path length can be given as:

$$\text{small tool path length} :=$$
$$\frac{Area\ of\ Face - Big\ tool\ Area\ removed}{small\ tool\ diameter * (step-over)} + boundary\ curve \quad (8)$$

Similar to the large tool case, Equation

$$\text{small tool path length} :=$$

$$\frac{Area\ of\ Face - Big\ tool\ Area\ removed}{small\ tool\ diameter * (step - over)} + boundary\ curve$$

(8 gives the planar path length of the small tool. However, most cutting circumstances require multiple depth steps, or z-steps to reach the required depth. Since each cutting simulation will have a prescribed Depth of Cut (DOC), we can determine if, and how many, z-steps are required. The complication is that the depth of the partition is unknown. The only information available is the volume of the partition and the area of the face.

3.4.6 Total Tool Travel

Using the volume of the partition and the face area, we can estimate the depth of the partition. Immediately, the concern is raised that this is only accurate for prismatic partitions. While this is true, for non-prismatic shapes, the incorrect calculation of the depth will still lead to a good estimation of total tool travel required.

Given depth, we can determine how many Z-steps are required, given the DOC determined by the tool diameter. These calculations are performed as follows:

$$\text{small tool path length} :=$$

$$\frac{Area\ of\ Face - Big\ tool\ Area\ removed}{small\ tool\ diameter * (step - over)} + boundary\ curve \quad (9)$$

$$N_{z\ steps} := Ceiling\left(\frac{partition\ depth}{DOC}\right) \quad (10)$$

The ceiling function is required since the depth is likely to not be a perfect multiple of the DOC, yet we need an integer value for number of z-steps. The result is that the final pass will have a smaller DOC than all the previous passes, which reflects the approach CAM software takes with multiple z-steps.

Similar to the adjustment taken to add path length to account for the jump between offset curves in plane, an adjustment must be taken to add path length for the move between z-steps. Like the planar case, the move between z-steps is taken to be a path at a 10 degree angle down from the plane, until the next plane is reached.

For both a big tool and a small tool, if required, we have determined the planar tool path length, the number of z-steps required, and the required tool motion adjustments. At this point we can calculate the total tool path length for both of these tools, which the machining model will use to determine machining time.

large tool total path length :=

$$\left(\sum L_{off}.big + \frac{R_{tool}}{sin(20\ degrees)} * \left(N_{offsets} - 1\right)\right) * n_{z.big}$$

$$(12) + \frac{R_{tool}}{sin(10\ degrees)} * \left(n_{z.big} - 1\right) \qquad (11)$$

Where L is the length of all offsets selected for big tool path, R is the radius of the large tool, N is the number of planar offsets selected, and n is the number of z-steps required. The total path length for the small tool is given by Equation (12.

small tool total path length :=

$$\left(\frac{Area\ of\ Face - Area\ removed\ by\ large\ tool}{R_{smalltool}}\right) * n_{zsteps} + \ boundary\ curve \qquad (12)$$

Where the area removed by large tool is determined by Equation 8, and n, the number of z-steps, is calculated using Equation (10.

3.4.7 Machining Simulation

For milling operations, the operator must select the speed of the machine (RPM), the feed of the machine (IPM), the step-over (in) and the depth of cut (in). While RPM and feed of the machine are usually chosen by the machinist based on past experience, there are fundamental machining mechanics that drive what these values should be; cutting speed and feed per tooth. For the work presented here, a table containing values for Cutting Speed, Feed Rate, and unit power for a variety of materials is used. It is populated using published data from Walker [8] and Kibbe [9]. The data in this table is loaded by the software at the beginning of the process, to be accessed by the machining model when needed.

The Cutting Speed is a function of the part material and the tool material. Cutting speed can be translated to a machine RPM if the tool diameter is known using Equation

$$RPM := \frac{CS*4}{D_{tool}} \qquad (13.$$

$$RPM := \frac{CS*4}{D_{tool}} \qquad (13)$$

The IPM is the linear advancement of the cutting tool, and is given in units of inches per minute. It is based on feed per tooth, a published value depending on the tool and part materials and tool diameter, and the RPM of the machine. The IPM can be calculated using Equation

$$IPM := FR * N_{flutes} * RPM \qquad (14.$$

$$IPM := FR * N_{flutes} * RPM \qquad (14)$$

Step-over and Depth of Cut are functions of part and tool material, and tool diameter. For the work presented here 50% is used consistently for both step-over and DOC, although this parameter is very easily adjusted to match machinist preference.

The Material Removal Rate (MRR) is an important machining parameter that determines the power requirement of the machine, and can be found using Equation $MRR := DOC\ (in) * Stepover\ (in) * IPM\ (\frac{in}{\min})$ (15.

$$MRR := DOC\ (in) * Stepover\ (in) * IPM\ (\frac{in}{\min}) \quad (15)$$

The power required to achieve a specific MRR is a function of the material being cut, characterized by the materials' Unit Power, a published value. The power requirement can be calculated using Equation (16.

$$Spindle\ Power := MRR(\frac{in^3}{min}) * UnitPower\ \left(\frac{HP*min}{in^3}\right) \quad (16)$$

At this point, the machining parameters have all been determined, but they still need to be checked against the capabilities of the specific machine used in this operation. We can directly compare RPM, IPM, and HP against the abilities of the machine, but how can these values be corrected if there is a limitation? Comparisons are implemented in the method to check a parameter immediately after it is calculated, to limit error propagation. The checks for the three relevant parameters are given, in order of application, below.

1. If the RPM exceeds the machine maximum, the cutting speed is reduced until the RPM value is acceptable
2. If the IPM exceeds the machine maximum, the feed per tooth (FR) is reduced until the IPM value is acceptable (Kalpakjian [10], El-Hofy [11]).
3. If the spindle power exceeds the machine maximum, reduce feed per tooth (FR) until the HP value is acceptable.

Although literature suggests the reduction of FR to lower power requirements, augmenting the step-over, DOC, CS, or RPM is being investigated as an alternative solution.

Given the IPM setting of the machine, and the tool path length for the given tool, we can determine the machining time for that tool. Naturally, the entire model must execute twice if two tools have been selected for the partition. The total machining time can then be expressed by Equation (17.

$$Time_{machining} := \frac{Large\ tool\ Path\ length}{IPM_{large}} + \frac{Small\ Tool\ Path\ Length}{IPM_{small}} \quad (17)$$

The extra time associated with using two tools, (i.e. tool change time) will be accounted for when the non-productive production times are determined in Section 3.4.8 of this work.

3.4.8 Non-Machining Production Time

For any machined part, particularly if it requires multiple processes to complete, the actual machining time does not accurately depict the time required to create the part. Non-productive times can account for as much as 66% of the time (Enparantza [12]). Luckily, since AMFA follows a part through the entire process plan, we have the opportunity to capture and quantify these non-productive part processing times, which occur between operations.

Along with the machining time estimation, for each operation, the evaluation method will also compare pertinent information for the current operation, and the previous operation. There are a number of parameters that may change between operations, which can each be represented as non-machining production times. These times are broken into three categories: fixture time, tool change time, and machine switch time.

Fixture time is the amount of time required to fix a part in a machine so that it may be machined. This value can vary greatly, based on part size and complexity, as well as the method of fixturing. At this time we are unable to predict these parameters based on part and facility data, and are using a fixed time value for fixturing. Future efforts will focus on expanding this functionality. There are two checks that need to be performed to determine the fixture time: first operation or orientation changed.

The first operation simply checks to see if this is the first operation that is being performed in the process plan for this part. If this is true, the fixture time is assigned, since the part will always need to be fixtured. If it is not the first operation, we simply check to see if the machine or orientation has changed since the previous operation. If the part has been moved to a different machine, or it is on the same machine but has been re-oriented, it is assigned another time value, composed of summing an unfixturing time, inspection and deburring time, and a fixture time. A flowchart of this process can be seen in Figure 26:

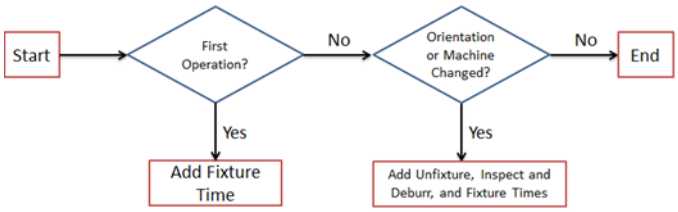


Figure 26. Flowchart of Fixture Time Assignment

Tool change times will contribute only a small amount to the overall process plan time and cost, since many modern CNC machines have tool change times of only a few seconds. However, for completeness, each tool change and the corresponding time delay in processing is recorded into the process plan. The tool change time can become more important when dealing with a machine

that does not automatically change tools, as the time for an operator to perform the switch could be greater than a minute. Recall that the tool selection method is able to select two tools for a single operation. If this is the case, we will add the tool switch time of the machine to the operation here, in the tool switch time method. The tool switch time of the machine is also added when the machine or tool is different from the previous operation, to account for the extra time in switching operations due to setup of the tool in the machine.

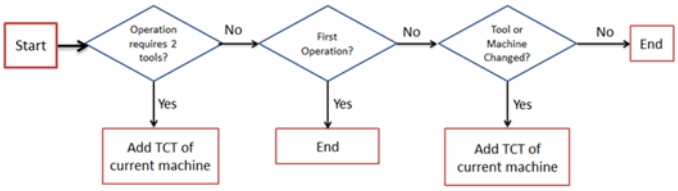


Figure 27. Flowchart of TCT Assignment

The machine switch time module is intended to capture the effect of the shop floor layout, where it takes time to move the part between machines required for consecutive operations. Currently this is a single value assigned when the current machine is different than the previous machine, since we are not working with any particular layout of machines. If the layout were known, this time would be a function of the walking distance between machines, and the size/weight of the part. Currently we are researching this method to include the parameters previously listed.

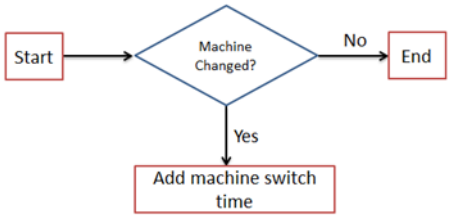


Figure 28. Flowchart of Machine Switch Time Assignment

3.4.9 Determining Operation Cost

At this point we have accounted for all of the time periods required to make the part. To translate this list of times to cost, the rates for the shop that is making the part are required. Currently these are defined as machine rate and operator rate. Machine rate is intended to be the cost charged to operate a machine, free of operator. The operator rate captures the charge for the labor performed by the shop floor staff. This includes the time they spend on machine setup, inspection, etc. For many shops these may be a single value, as an hourly charge for however long it takes to make the part. We have chosen to keep them separate, to allow for shops that choose to charge different rates depending on what machines are required and how much operator time is involved. We can now take these two cost rates, the machining time, fixture

time, tool change time, machine switch time, and calculate the total cost to perform the given operation, using

$$Operation\ Cost := Rate_{machine} * \left(t_{machining} + t_{toolswitch}\right) + Rate_{operator} * \left(t_{fixture} + t_{machineswitch}\right) \quad (18)$$

Equation (18) will estimate the total cost of an operation, including non-machining times taking place between the current operation and the previous operation. Once an entire process plan has been generated, the cost of each operation in the plan can simply be summed to estimate the total production cost of the part.

3.4.10 Tolerance Analysis

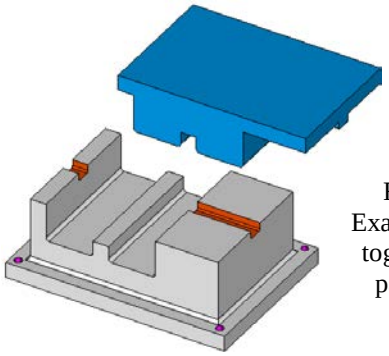
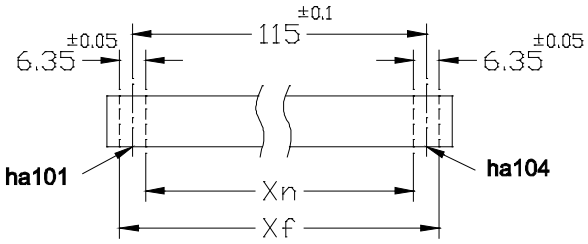


Figure 29. (LEFT) Example of a part (gray) together with a mating part in the assembly (blue)




```
! Assembly conditions are defined here.
! It includes pair of faces to do tolerance analysis
! and its design intent dimensional limit between them.
!
1  ha35   ha53   88.8   89
2  ha53   ha69   38     38.2
3  ha47   ha35   38     38.2
4  ha101  None   6.3    6.4
5  ha104  None   6.3    6.4
6  ha101  ha104  121.2  121.5
7  ha101  ha104  108.5  108.8
```

Figure 31. (ABOVE) For a set of pairs of faces the dimensional tolerances that are permissive.

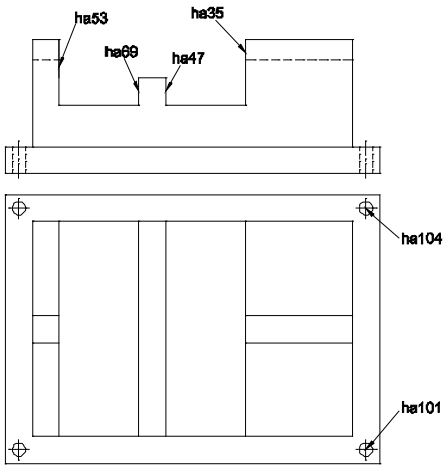


Figure 30. (RIGHT) Labeling faces on the part over which tolerances are specified.

In addition to the time and cost to manufacture a part, it is also important to determine the probability with which a chosen process will produce parts that meet specified design tolerances. The figures above show an example of a part (gray) together with a mating part in the assembly (blue). The diagrams to the right label certain faces on the part over which tolerances are specified. The text file on the left specifies for a set of pairs of faces the dimensional tolerances that are permissive.

3.4.11 Dimensioning, Tolerancing and Variability Analyses

Towards the end of the period of performance on the initial iFAB effort led by PARC, two additional Tasks were added under the leadership of Arizona State University (ASU). First, the team conducted a Geometric Dimensioning & Tolerancing (GD&T) case study on a particular subassembly design generated by Vanderbilt's META tools. Second, the team developed a set of software modules to demonstrate a streamlined procedure for variability analyses of candidate process plans.

Vanderbilt's GD&T scheme for a partial subassembly was used. The specifications were sent as a STEP file (geometry) but GD&T could only be sent in the form of screen scans (jpeg files). The main assembly consisted of a Chassis, Roll-bar set, Rear bracket and Radio boxes. This assembly was divided into two independent pseudo assemblies for tolerance analysis. The contributing

dimensions and features (tolerance chain) with pre-existing tools were determined. Based on sensitivities and percent contribution to variance, an alternative GD&T scheme was suggested to improve manufacturability. The original GD&T scheme was found to be incomplete. Few size dimensions had been explicitly specified, presumably left to be extracted from CAD which makes the direction of control ambiguous. No basic dimensions were given for position tolerances. There were no inter-datum controls in the datum reference frames (DRF). Only those specs essential for analysis of the as-is design communicated by Vanderbilt were added. Two extreme conditions of assembly between Chassis and Roll bar set were analyzed. One extreme condition is when the position and size tolerances are such that the pins in Roll bar set are nearest to each other while holes in Chassis are farthest to each other, and vice versa for the second extreme condition. 1D charts were supplemented with Gaussian distributions to compute statistical acceptance rates. 2D linearized analysis was conducted with a commercial CAD system. 3D Monte Carlo simulation was done with the ASU GD&T Testbed. Under the existing GD&T scheme it was determined that the acceptance rate would be below 50%. By changing the GD&T scheme, using better datums and a more efficient datum flow, it was demonstrated that the acceptance rate could be improved to 96%.

A set of modules for enhancing AMFA output were created, extracting data relevant to variability and transforming gxml and json formats to Constraint Tolerance Feature format (CTF). Loop detection, 1D chart construction, 1D statistical analysis, and 3D Monte Carlo simulation were implemented to analyze manufacturing dimensional variability of process plans, without third part libraries or packages. Modules created were written in C++. For a sample part, the team produced a multitude of process plans and the entire extraction, transformation and analysis process was demonstrated.

In order to do manufacturing variability analysis on each plan, it was necessary to add workholding information and tolerances for each operation in each plan. A template was created to associate these attributes in the json file based on feature and operation types. The original json file only identified a bottom support face in set up information. Two additional faces needed to be identified for workholding so they could be used as secondary and tertiary datums in variability analysis. The json, “fixed face” entity was expanded to include up to three mutually orthogonal faces. Another enhancement in json needed for this project was a listing of “exposed faces” or faces created by each machining operation. Fixed faces and exposed faces were required to have tolerances specified. Tolerances needed depend on the feature type and operation type. There are many factors that contribute to manufacturing variations: machine accuracy, process parameters (feeds, speeds), workpiece material, tool wear, fixturing, etc. A number of theoretical and empirical causal models of machine tool errors can be found in the literature. Since the purpose of AMFA is manufacturability analysis and not detailed process planning, we proposed a simplified model for determining variation values.

The variability analyzer needs to know the particular variations on the part that are critical to assemblability. A format was developed for specifying this information (assembly conditions). Eventually, this information needs to come from the designer, but at this time it is manually input.

It is envisioned that 3D plots of process plans (time, cost, quality or acceptance rate) can be generated to select best plans or to identify opportunities for improvement.

Due to the short duration of this part of the total effort the scope was limited to 3 axis milling. Because milling produces largely prismatic parts, run-out tolerances were not in scope. Free form profiles are not supported at this time. Only one sided clearances are considered; floating clearances are not included. Machines and cutting tools are hypothetical. In the future, accuracy tests on actual machine tools to be used in the AVM program.

3.5 Manufacturability Design Feedback

By manufacturability design feedback, we mean a method that either guarantees that a part or assembly can be manufactured as the designer specifies, or a method that suggests to the designer what aspects of a part or assembly cannot be met given the constraints of a foundry. At a high-level, this feedback is provided to the designer in one of two ways: Design for Manufacture (DFM) Rules, and Relaxation Search.

3.5.1 DFM Rules

In mechanical design there are standard best practices and rules of thumb generally referred to as DFM best practices. These rules and guidelines tend to be specific to a high-level manufacturing process, such as rules for milling and drilling, versus rules for casting and molding, etc. Before exploring possible process plans, a submitted part is first tested for violations of any DFM rules. The results are presented graphically in a GUI by highlighting faces that raised flags in particular DFM tests. The designer can then modify their part accordingly and resubmit.

The motivation for explicit DFM checks is that some features of a part may make that part very costly to manufacture, likely to have tolerance issues, or may even render a part completely non-manufacturable by a class of processes. For example an internal pocket with sharp corners is non-manufacturable by any milling process, while drilling on an inclined face is feasible but the risk of the tool bending and walking is very high.

For milling and drilling, DFM checks are done for: sharp corners and edges; obstructed holes; curved, blind holes; drilling on a slope; holes with flat bottoms; and thin walls.

3.5.1.1 Non-Manufacturable Sharp Corners and Edges

For a milling process to generate a sharp edge at the intersection of two planar facets that meet at an angle of less than 180 degrees, an end mill must be used, as opposed to a ball mill for example, and furthermore the tool must approach the faces that meet at that edge either along the surface normal of each face in the direction perpendicular to the surface normal and perpendicular to the edge for each face (the axis of the tool laying on the surface of a face and its end tracing along the edge). The test then is to check whether the edge is obstructed by any other feature of the part along the tool's approach direction; if the tool cannot approach the edge then it is non-manufacturable. Furthermore, if three or more such edges meet at a corner, such as the sharp corners in the pockets of the part in Figure 32 then the vertex in non-manufacturable by any milling process. In Figure 32 a number of faces are non-manufacturable because of obstructed sharp edges and non-manufacturable corners (marked in red), and some faces are manufacturable but restrict the approach direction of the tool (marked in yellow).

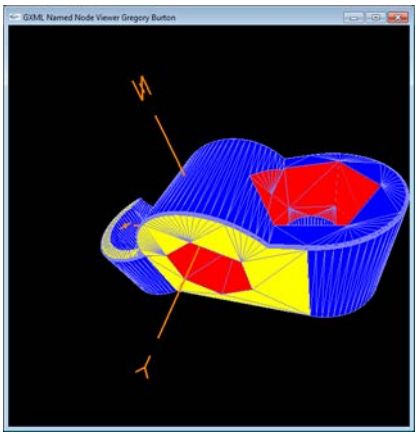


Figure 32. Non-Manufacturable (red) and Restricted Manufacturable (yellow) Faces

3.5.1.2 Obstructed Holes

Similar to the test described in the previous section, a hole must be approached by a tool from a limited set of directions; two directions if it is a through-hole and one direction if it is not. The test then is to see whether the approach direction is obstructed by any other feature of the part. In Figure 33 some holes are manufacturable, but from one specific approach direction (yellow), and other holes are inaccessible (red).

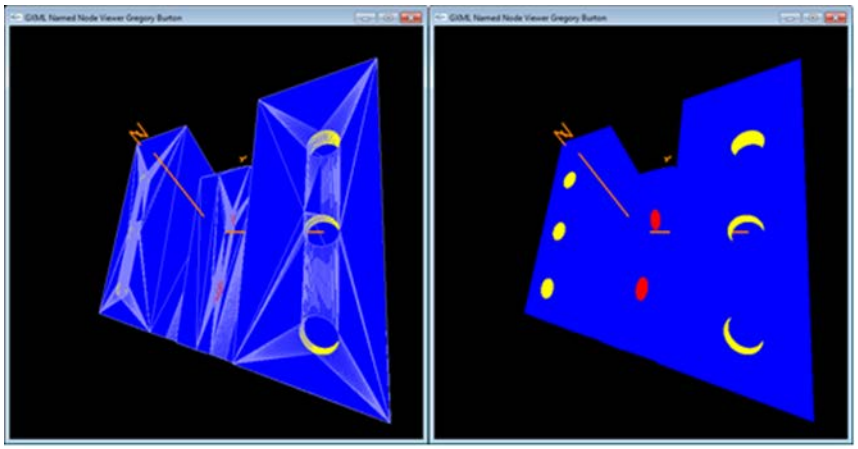


Figure 33. Non-Manufacturable Obstructed Holes (red)

3.5.1.3 Curved, Blind Holes

A curved, blind hole is non-manufacturable by any milling process. In this case the existence of such a feature is all that has to be tested.

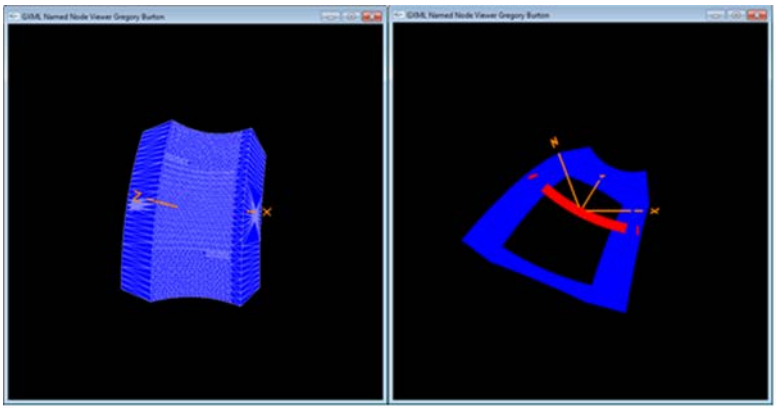


Figure 34. Non-Manufacturable Curved, Blind Hole

3.5.1.4 Drilling on a Slope

A hole drilled on a slope is not a non-manufacturable feature, however, because of the high likelihood of the tool bending and walking when it initially touches the surface of the stock, it represents a tolerance issue. In this case the existence of such a feature is all that has to be tested.

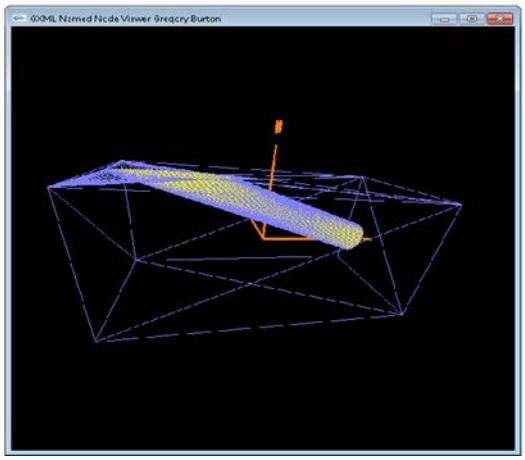


Figure 35. Difficult to Manufacture Hole with a Flat Bottome Drilled on a Slope

3.5.1.5 Holes with Flat Bottoms

Though not strictly non-manufacturable, a hole with a flat bottom is difficult to manufacture. If the hole is to be threaded it is non-manufacturable. It is better practice to allow space for the bottom of the tool, such as a cone if it is to be drilled. In addition to the hole drilled on a slope in Figure 35, the flat bottom of the hole is also flagged.

3.5.1.6 Thin Walls

If the walls of the stock are required to be cut less than some material-specific thickness then the material may bend or deflect under the force of tool. We measure interior wall thicknesses and flag them if they are less than some cutoff parameter.

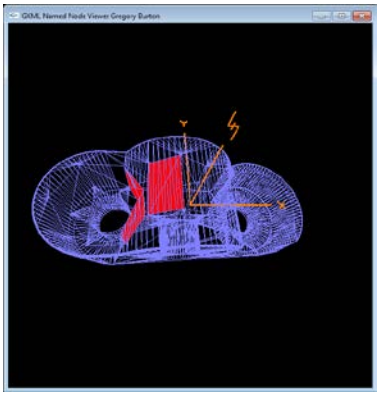


Figure 36. A Part with Very Thin Walls

3.5.2 Relaxation Search

If the search for a process plan comes back negative, i.e., no plan could be found, then a secondary search, called relaxation search is performed. In this search, changes that could be made to the non-manufacturable design to make it manufacturable are identified. Intuitively, this search takes liberties as it explores the possibilities for manufacturing the design.

Recall that the search for process plans is based on graph grammar rules that describe the possible operations that can be performed on a graph describing a geometry. Roughly, these rules correspond to manufacturing steps, such as removing a certain sub-volume using a specific machine and tool, for instance milling a pocket with an end-mill on a vertical machining center (CNC machine). Each such rule consists of a left-hand-side (LHS) and right-hand-side (RHS). The left-hand-side, as a reminder, captures the conditions under which the rule applies, i.e., the operation can be performed. The right-hand-side then describes the result of the operation, which for the purpose of this section can be thought of as just stating that a certain partition has been removed/machined.

The novel concept of relaxation search relaxes conditions found in the left hand side of a rule, in order to make rules applicable where they are otherwise not. Relaxation search keeps track of all these relaxed conditions -- if relaxation search is successful, these relaxed conditions describe possible changes that could be made to the desired geometry in order to make it manufacturable. Relaxation search aims to find the minimal set of changes to the given graph, in order to still find a solution.

The relaxation search operates as follows:

- If the part is non-manufacturable:
    - sub-volumes that could not be removed are identified
    - determination is made for each machining capability (e.g., end-milling):
- why it was inapplicable (for example: sharp inside edge)
- how the graph can be changed to make it applicable (e.g., add a fillet)
- what the resulting 3D design changes are

3.5.2.1 Relaxation Search Example

As an example, consider Figure 37. In the shown part there is a number of joining sharp inside edges. These edges make the part non-manufacturable using only a mill, because inside edges can only be milled if they are accessible from both sides, or they have a fillet.

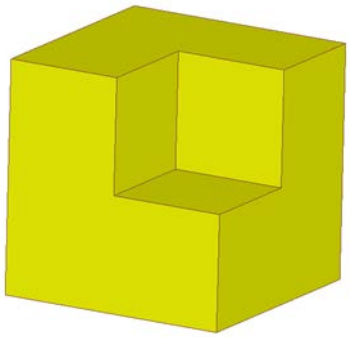


Figure 37. Sample Non-Manufacturable Part

As a result, when presented with this geometry, or any geometry that contains such a feature, the search for a feasible process plan for this partition is going to fail. This triggers relaxation search. Relaxation search finds that, among other rules, the end-milling rule is not applicable, because there are inside edges of the pocket that are not marked as tangential. This means that at least one of the transitions from one face to another in the pocket is not smooth, there is no fillet. Relaxation search now conjectures that if this lack of the label tangential on one of the edges is resolved, the end mill rule would apply, and search for a plan could continue. In the example, since this is the only partition to be removed, the search would succeed afterwards, with no more relaxation necessary.

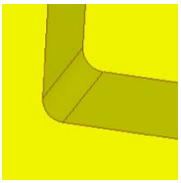


Figure 38. Recommended Design Change: Addition of a fillet

As a result, the search will produce a process plan that uses the end mill rule despite the violated condition, and points out to the user that this is only a viable plan, if the required changes are made to the geometry. These changes are represented as a textual description, and they correspond to the change shown in Figure 38, there a fillet has been introduced in the place of a sharp inside edge.

Conceivably, the space of possible design changes is infinite. In order to still be able to make practical progress and demonstrate value, in our implementation the relaxations that we consider are domain independent but limited to only adding or removing labels to or from the graph representation of the geometry. This means that relaxation search considers properties of nodes, arcs, and hyperarcs when conjecturing modifications, but does not conjecture adding or removing such entities (nodes, arcs, hyperarcs). In practice, this certainly limits the space of changes being explored, but already in itself presents a huge space that needs to be considered. Pragmatically, it seems that most modifications regard properties captured by labels. Nodes, arcs, and hyperarcs are more closely related to the rough shape as opposed to its details. This means that adding or removing such entities might not be meaningful and in fact result in physically infeasible shapes. On the other hand, labels mostly capture properties that certain aspects of shapes may or may not have and as a result are often good candidates for changes.

By virtue of the developed relaxation search being principled and founded in a formally theory, it allows for a variety of uses in exploring the design space. Specifically, it does not rely on hand-coded relaxations or design change templates, but can be used with any future implementation where additional machining operations may be encoded, not foreseen at this point of the project.

3.6 Indifference Curve

In order to make trade-offs we implemented a mechanism for learning the indifference curve of the designer when several metrics exist. When several possible process plans for manufacturing a given design exist the designer is presented with a number of non-dominated solutions to choose from. Given the choices made, the mechanism incrementally learns to trade off the metrics according to the estimated, true preference of the designer. This is used to guide the heuristic search to explore the most promising solutions first.

From the designer’s choices among those solutions, the designer’s indifference curve is learned. The indifference curve is then used to determine the optimal process plan, according to the designer’s individual preferences. In the case where there are features in a component that cannot be manufactured using any of the machines in the foundry, a relaxation search process starts.

The user’s preference function was modeled as $P(c,t) = wt + (1-w)c$, where c is cost, t is time, and w is a weight representing the tradeoff between time and cost as preferred by the user. Our solver begins with w in the range of [0,1], and through repeated interactions with the user, learns to narrow down w to a single value. Every selection made by the designer gives us a coordinate pair (c,t), which is then used in conjunction with our preference function P to constrain the possible values for the free variable w. This then completely defines the preference function P(c,t). As we narrow down on the user’s preference function, we use this information to prune process plans.

4.0 RESULTS AND DISCUSSION

4.1 Characteristics of Implementation and Deployment

AMFA is a fully automated feedback tool. It has been deployed as a web-based application that currently runs on a private cloud at PARC. The computational performance was tuned and the computation parallelized to achieve a significant speed up. The graphical user interface (GUI) was written entirely in HTML5 [13] and JavaScript [14] using various open-source libraries including Sencha's ExtJS [15], and the Three.js WebGL library [16].

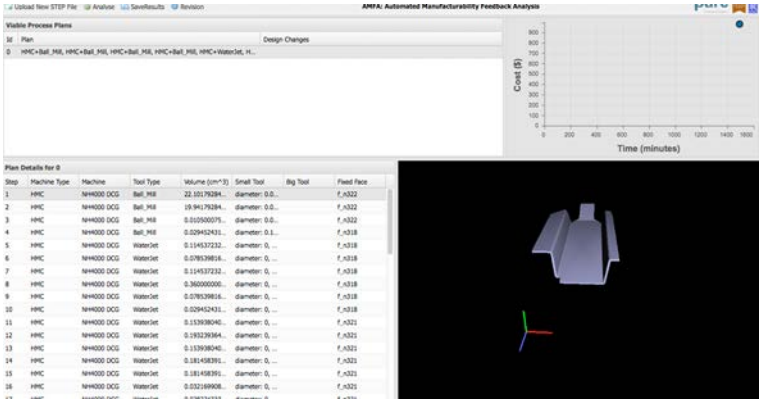


Figure 39. AMFA GUI Web Page

Some of the other features of AMFA which may be important to the continued success of the AVM program are as follows:

- AMFA is customizable based on available resources of a particular foundry. We have demonstrated that AMFA can work with manufacturing machine libraries provided by various entities (in this program it was demonstrated with libraries from both Boeing and GATech).
- AMFA produces estimates of manufacturing time and cost.
- AMFA is capable of learning designers preferences over manufacturing metrics (time vs. cost) to iteratively generate plans that are closer to the real user preference between such competing metrics.
- AMFA admits customization of the search algorithm via numerous parameters that can be set in the command line or the CGI GET query string. This allows the search to be configured to be, e.g., be faster but result in lesser quality plans, or a little slower but produce more optimal plans.
- The AMFA GUI shows recommended design changes to the designer to improve manufacturability when a non-manufacturable part is uploaded.
- AMFA allows the user to step through the individual steps of a proposed plan and see the staging models: the geometry the way it looks after any prefix of the plan has been

executed. This feature allows easy validation of a plan, and provides a concrete means for designers to communicate with manufacturing engineers about the feasibility and pros and cons of feasible plans.
- AMFA allows the user to save computed result for future reference.
- Performance tuning increases AMFA performance by a factor of 20x to 30x on parts that include a lot of features.
- Parallelization of AMFA further increases performance by a factor of 33% on a quad-core system.

In summary, AMFA produces near-real time manufacturability analysis results.

4.2 Manufacturing Processes Handled

Currently the AMFA tool is mainly focusing on the 3D axis machining process. Most of the traditional manufacturing operations that can be implemented in 3D axis machines are encoded into the tool. They include:

- Milling

Both end milling and ball milling are handled. While end milling is intended for use in most cases, such as profile milling (pockets, holes, slots, chamfers, etc.), face milling, tracer milling or plunging, ball milling is used specifically for creating inner fillets for a given part.

- Drilling

The drilling operation in AMFA is able to produce both through holes and blind holes. It is a highly generalized manufacturing process that indicates all possible regions (typically holes) in a given part where a drill bit can be used.

- Countersinking

This is a specific finishing process for the countersinks sitting on the holes. It represents a manufacturing process to produce countersinks with specially designed tools.

- Counter-boring

In AMFA, this manufacturing process is encoded into milling operations. For the counter-bores, instead of using specific tools, AMFA uses current milling tools to machine them.

- Bending

This is a non-material-removal manufacturing process. AMFA implements this operation by first detecting all regions that need to be bent, then calling bending operation to take care of these regions.

- Welding

For a given final part, AMFA detects if the welding operations have ever been used to manufacture it. If such operation is detected, an unbending operation will be called to separate the welded parts, and each separate part will then be manufactured using the operations mentioned above.

- Manufacturing processes from a cast part

If a part needs to be machined from a cast raw material, AMFA will use the cast model and the input solid model to compute the regions to be machined. These regions will be manufactured using the operations mentioned above.

A few non-traditional manufacturing processes are also built and being built into AMFA.

- Sheet metal cutting (Water-jet cutting)

For sheet metal parts, instead of machining it traditionally, AMFA uses water-jet cutting to cut out the part profile.

4.3 Part Geometries That Can Be Processed

AMFA has been tested on a variety of parts provided from Rolls Royce and other iFAB teams. For these parts, there are a few constraints that must be satisfied. The constraints are summarized into several categories and a few screen shoots of the sample parts for each category are provided.

1) The parts should be able to be decomposed into prismatic partitions using the convex decomposition algorithm built in AMFA. The sample part Figure 40, although looks complex, is implementable by AMFA since all its partitions are prismatic.

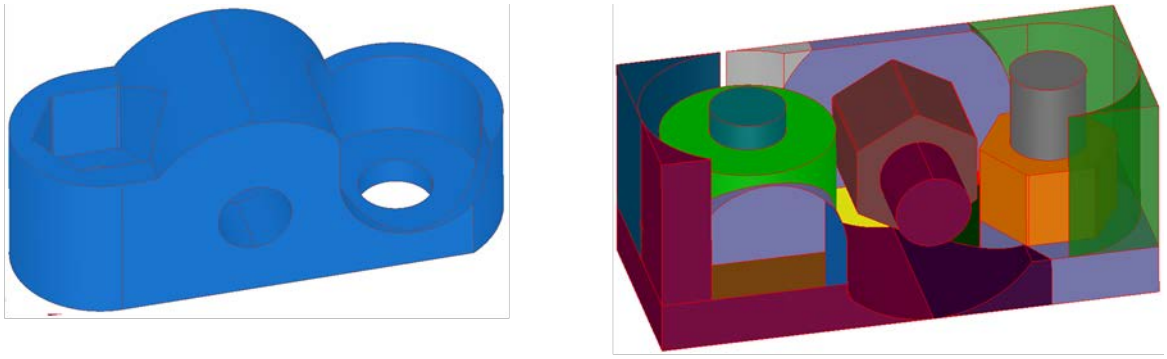
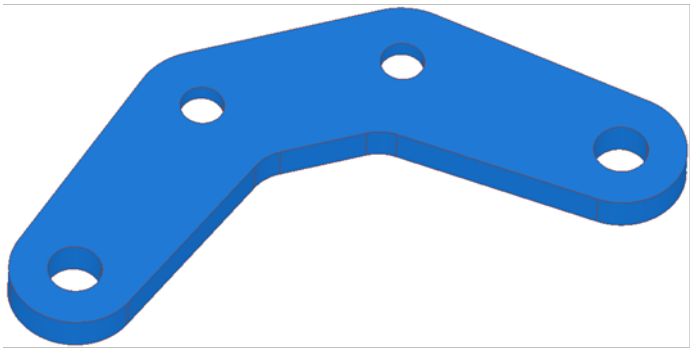


Figure 40. Sample Radiobox Component from META Design Challenge

2) There should be no fillets/round edges around the boundaries of the parts. Due to the limitation of the CAD kernel used in AMFA, small fillets/round edges will crash the software. However, since these features are machined in the final finishing process, removing them does not affect the generation of high level manufacturing recipes.



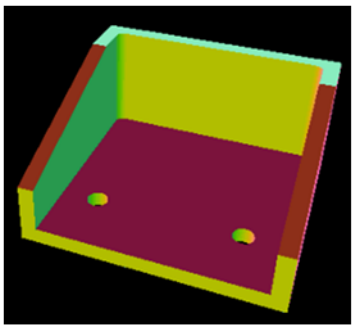
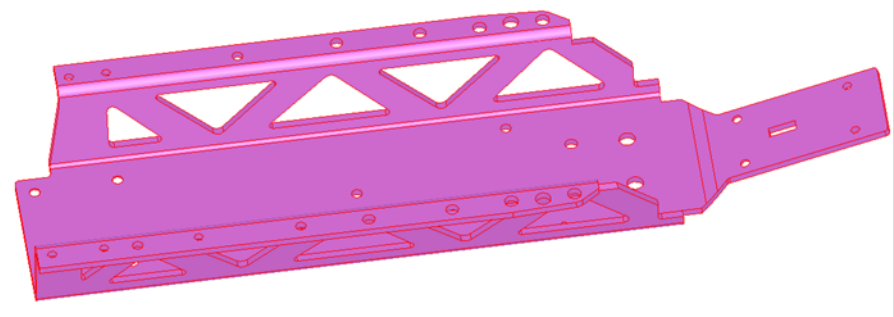


Figure 41. Sample Test Cases

3) All kinds of sheet metal parts are implementable in AMFA. Among these parts, some may need non-traditional operations, like bending and welding. AMFA will generate corresponding recipes for these operations. For example, AMFA will detect that the bending operations were used to manufacture the first sample part Figure 42. The corresponding recipes will then be generated automatically.



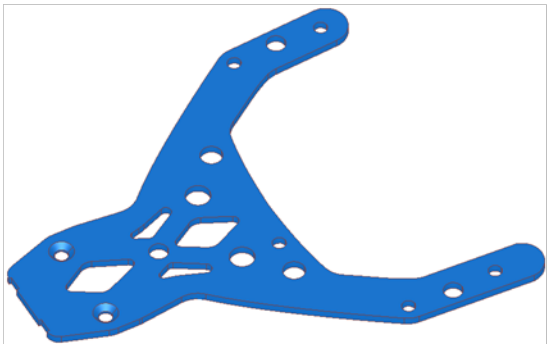


Figure 42. Additional Sample Test Cases

4) Parts that need to be pre-cast. For the sample part Figure 43, if the user provides both the final part and the cast raw material, AMFA will automatically compute the regions of removal material from the raw material.

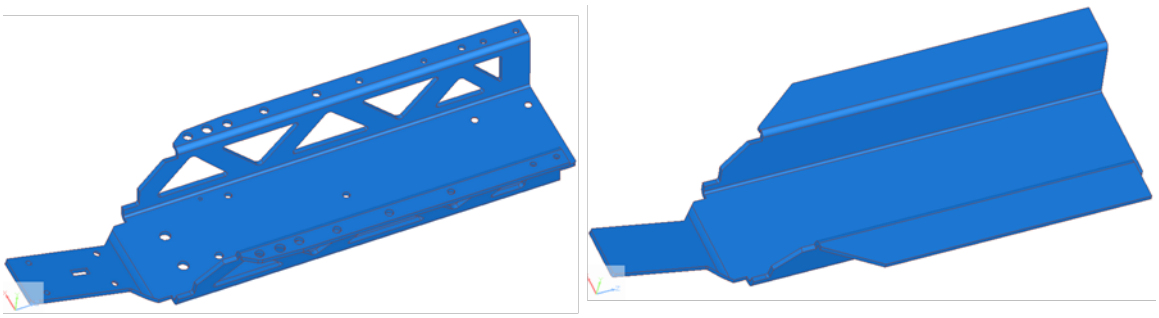


Figure 43. Sample of a Pre-formed Part

4.4 Validation

The Automated Manufacturability Feedback Analysis (AMFA) tool provides the ability to analyze the solid model geometry generated during the design phase against real-time manufacturing global databases so the designer is brought into the manufacturing realm during the design phase. The ability of AMFA to analyze the complete solid model and not just features of the solid model enables a complete analysis of the component via graph-grammar rule sets for specific disciplines, in this case subtractive manufacturing. Only the complete solid model analysis covers the complete component definition of the geometry. This represents an improvement over the feature based manufacturing methods which require additional effort and input to efficiently analyze a component against manufacturing constraints. As 3D solid model geometry has replaced 2D drawings partial geometry definition so has AMFA analysis replaced feature based manufacturing. In today's industry the 3D solid model is the complete definition of the part or assembly and is the media used to communicate the product definition in the marketplace.

AMFA's ability to analyze solid model geometry and determine feasible manufacturing process plans given a known set of machines and tools over a theoretical diversified global manufacturing supply chain and provide the designer understandable feedback on staging models, times and potential costs for the supplied part in a timely fashion through an easy to use interface was the goal of the project. The measure of validation of the AMFA software will be the comparison of the AMFA produced process plans against manually produced process plans created by Rolls Royce manufacturing engineers. AMFA must create multiple process plans that are feasible per manufacturing engineering judgment. Finally, AMFA must produce manufacturing process plans in a timely fashion, less than an hour turnaround time from upload of geometry.

4.4.1 Ground Vehicle Frame Design and Manufacturing Analysis

To enable the pilot the Rolls Royce engineering team created solid model assemblies of three different vehicle frames to simulate the design process for a military ground vehicle. The solid models defined the frame structures of the three vehicle designs and represented a jeep configuration, MRAP (Mine Resistant Ambush Protected family of armored fighting vehicles) configuration and a XC2V (Experimental Crowd-derived Combat Support Vehicle) modular

tubular, (light weight) concept vehicle configuration. The initial jeep frame configuration was created as a “working” parametric solid model but not exercised for this project. The remaining two vehicles were created in static parametric form since real time design iterations on geometry was not required for the iFAB project.

The three virtual vehicle frame designs represent potential FANG submissions. Great care was taken so that the solid models created for the three virtual designs closely matched existing military ground vehicle hardware to closely emulate potential FANG designs. No real military hardware was measured for the creation of the virtual frame designs. The only input for the vehicle design concepts was images from the Internet where vehicle hardware was exposed and innovation on the part of the design personnel.

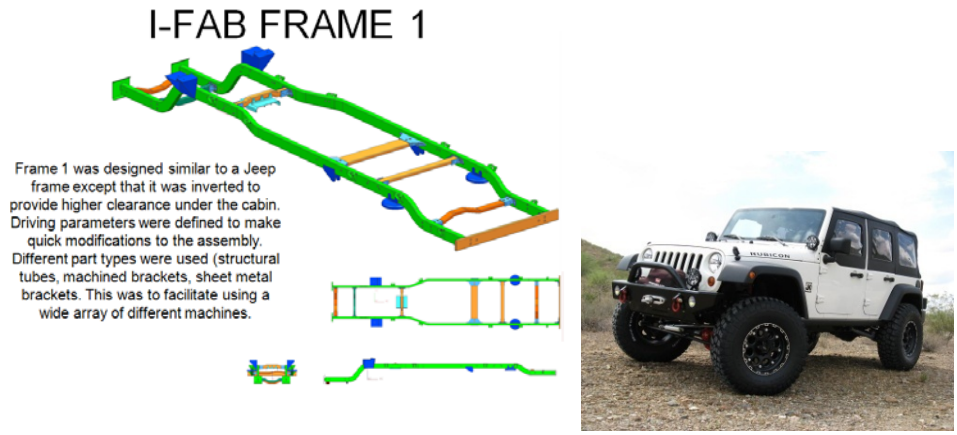


Figure 44. Jeep Frame Design Configuration

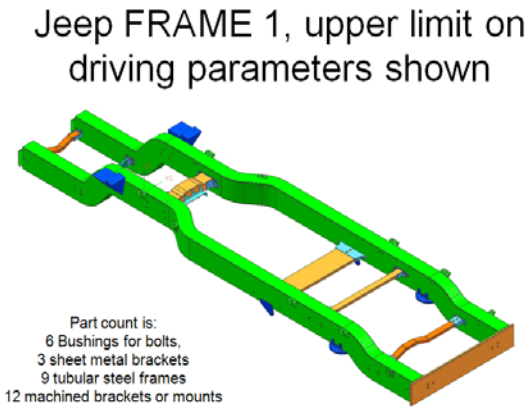


Figure 45. Solid Model Upper Range on Driving Parameters

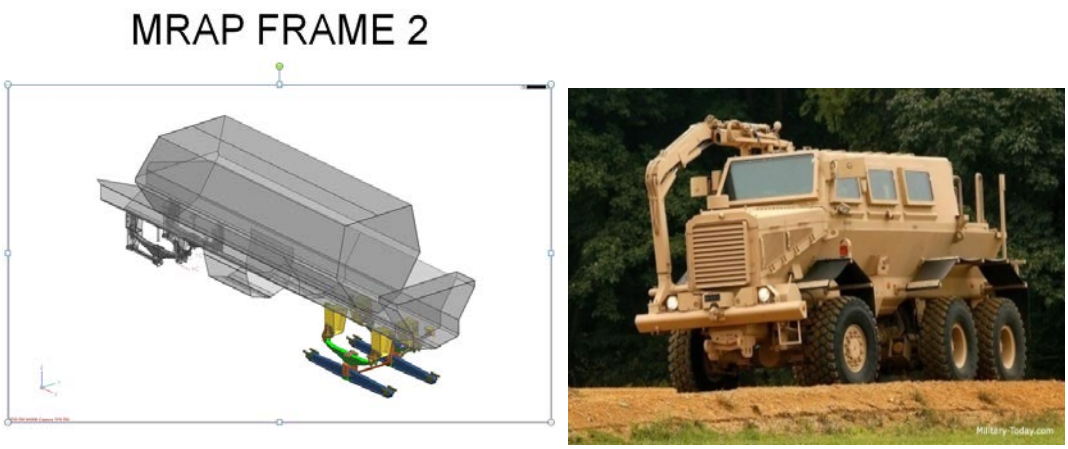


Figure 46. MRAP FRAME 2 Assembly and Production Vehicle

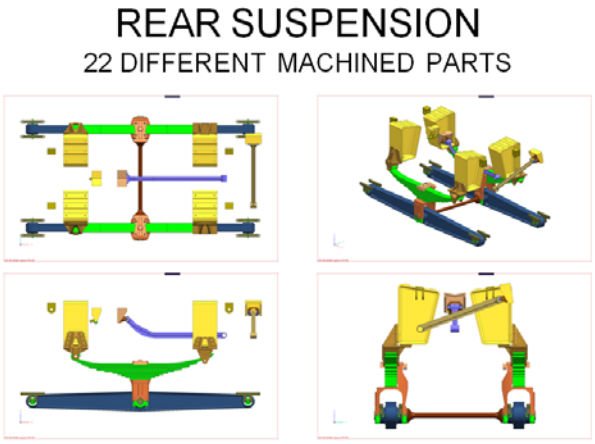


Figure 47. MRAP Rear Suspension Assembly

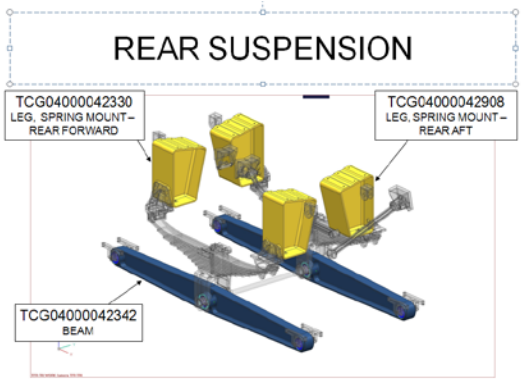


Figure 48. MRAP Rear Assembly View with TCG # Shown

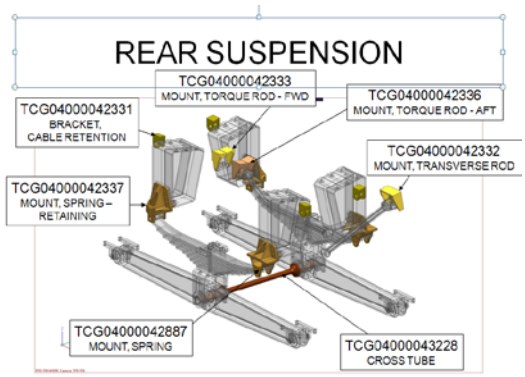


Figure 49. MRAP Rear Assembly View with TCG# Shown

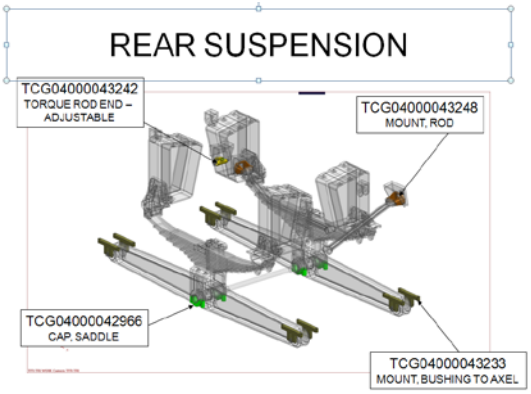


Figure 50. MRAP Rear Assembly View with TCG# Shown

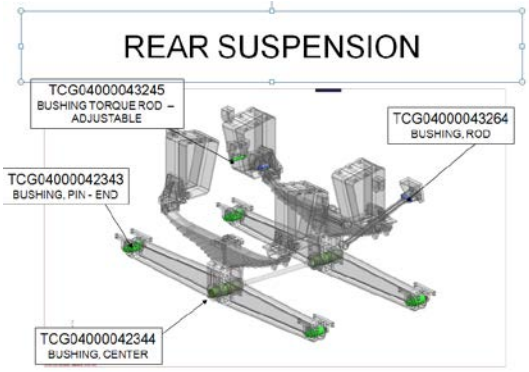


Figure 51. MRAP Rear Assembly View with TCG# Shown

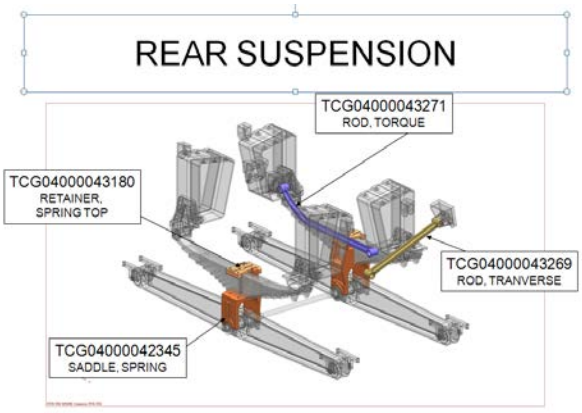


Figure 52. MRAP Rear Assembly View with TCG# Shown

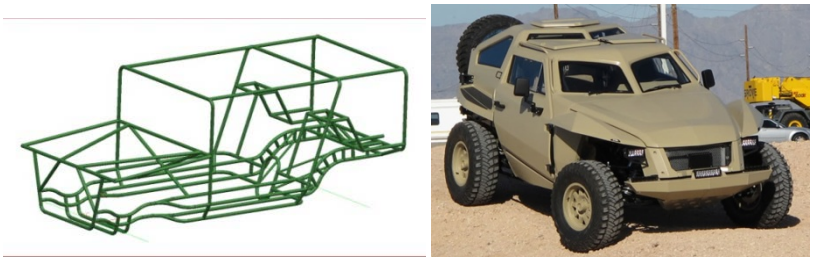


Figure 53. Frame 3, XC2V Concept Vehicle Style

4.4.2 User Testing

The validation of the AMFA software required comparison of manual process plans for test solid model geometries against AMFA generated process plans in order to certify the software methodology and functionality was working correctly. In addition, the AMFA software needs to read across several machine/tool databases to demonstrate the ability to connect to corporate Product Lifecycle Management (PLM) network databases that are used widely within industry today as well as provide an easy to use graphical user interface to facilitate designer interaction during the design process. The AFMA software will need to process incoming STEP geometry in a timely fashion, analyze the appropriate manufacturing processes available for the provided STEP geometry and then produce realistic preliminary process plans given the supply chain machines/tools available in the linked databases (supply chain). The AMFA technology demonstrator will be deemed successful if 1) feasible plans can be produced automatically, 2) the designer is provided usable feedback through an easy to use graphical user interface in a 3) real time fashion (solutions returned to the designer in an hour).

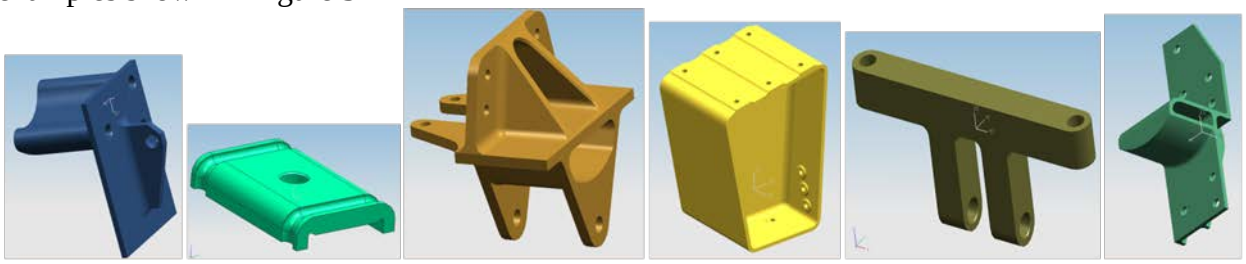
The essential advancement the AMFA software is to provide the ability to process a complete "volume" solid model STEP file provided by the design community in an automated fashion, process all of the features and the relationship to each feature in the topology as a whole so

Graph Grammar rules can evaluate potential manufacturing process plans given a known set of machines, tools and manufacturing techniques. The subtractive manufacturing technique will be reviewed in this body of work but the software team has demonstrated other welding and sheet metal manufacturing techniques that were impressive and further demonstrated the Graph Grammar solid model decomposition approach.

Seven Rolls Royce manufacturing and method development personnel conducted the testing of the AMFA software as well as provided consulting services for the software team. The Rolls Royce evaluation team was composed of two manufacturing engineers, referenced as A and B, two design methods personnel to represent the designers, two manufacturing managers and one manufacturing support engineer. The evaluation team provided manually created process plans as well as guided the development of the graphical user interface for AMFA via user recommendations and comments.

4.4.3 Test Cases

Numerous solid model geometries where created during the vehicle design phase with some examples shown in Figure 54



below. In the development of the AMFA technology it became apparent that large complex solid models would not be efficient in the shorten development cycle dictated by the project schedule. To expedite the development it was decided that a subset of test parts would be used to test AMFA software. The parts were chosen based on a few key features of the part to test the AMFA subtractive modeling rules.

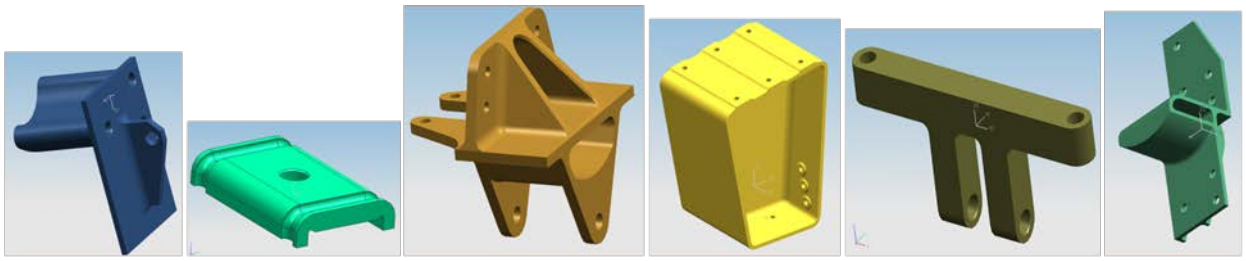


Figure 54. A sampling of example solid model geometries created during vehicle design process show.

The test case subset was made up of six example parts that were selected from the population of challenge parts contribute by iFAB participants as well as DARPA personnel. At the time of

selection the team was intending to include sheet metal forming so the chassis part, test case 5 – Figure 55, was included in the smaller test set. It was later decided to turn off the sheet metal forming rules in AMFA because it was not specified in the original deliverables to the government. The AMFA software at version 1360 detects the sheet metal part and reports to the user "the uploaded part either has non-manufacturable features, or features which cannot yet be processed by AMFA".

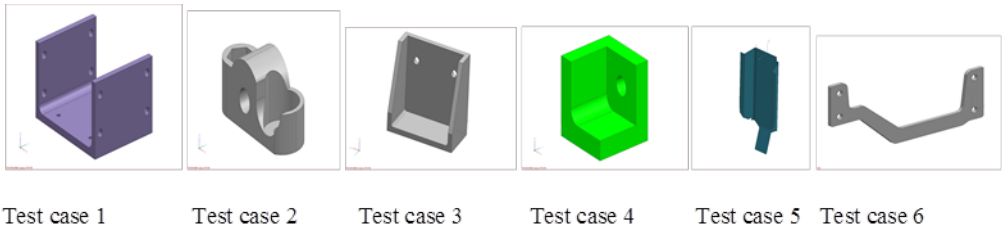


Figure 55. The six test cases used to test AMFA subtractive manufacturing analysis abilities.

Common 450 steel was selected as the base material for the six parts and a moderate tolerance was assumed for the manufacturing processes.

4.4.4 Testing Results

In total, seven Rolls Royce engineers provided testing and consulting services for the AMFA development. Two engineers were from methods development in the design community, two manufacturing managers, two manufacturing engineers (A and B) and one manufacturing support engineer participated.

The six test parts were provided to manufacturing engineer A and B. Two manufacturing engineers were used to show the difference in manually created process plans as well as provide an upper and lower bound on predicted times and process steps.

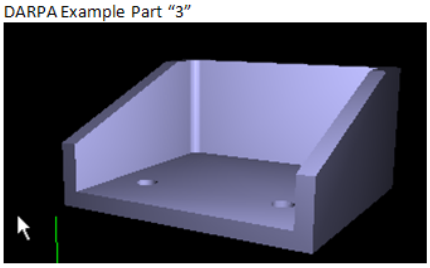
The manufacturing engineer A and B created preliminary process plans for the six components and detailed at a high level the manufacturing process steps required to produce the specific test parts. The manually created process plans were always slightly different between manufacturing engineering A and B, which was expected. Costs were not quoted by the manufacturing engineers for the purpose of comparison and would not be used as a critical measure. The cost structure for aerospace components is significantly different than general industry and would not be generated for the parts. The engineers would comment on the cost predicted by AMFA and highlight numbers that appear to be out of normal costs for general industry.

The importance of the staging models and the calculated machine times was deemed to be a better gauge of the AMFA software feasibility and accuracy.

If the staging process models created by AMFA matched a manual process plan it was deemed feasible. If the times range defined by the engineer A and B estimates overlapped the AMFA calculated process time range it was deemed feasible.

The high level manually created process plans for test case 3 from engineer A and B are shown in Figure 56.

DARPA Example Part "3"



Manual Plan "A"

Part: 03_test_TCG04*

	Operation	Machine	Tools Required	Set-Up Time	Door to Door (Processing Time)
10	Rough/Square	Mill 3Axis CNC	3	20	18
20	Rough/Semi-Finish/Finish Shape/Drill Bottom Holes(2)	Mill 3 Axis CNC	6	60	65
30	Deburr	Hand operation	3	0	10

All times are in minutes
1 off prototyping for initial run

173 minutes

Manual Plan "B"

03_test_TCG04

All work to be completed on a conventional 3-axis verical or horizontal machining center

		Load Time	Run Time	Set-up time
Oper. 10				
	Mill bottom face - hold in vise	1 min.	1 min.	30 min.
	Spot drill 2 places		1 min.	
	Drill 2 places completely through part		2 min.	
	Side mill all 4 sides		10 min.	
Oper. 20				
	Mill angled face - locate on pins in previous drilled holes using vise - deburr previous operation before loading	2 min.	1 min.	30 min.
Oper. 30				
	Spot drill 2 places - locate on pins in previous drilled holes using vise - deburr previous operation before loading	2 min.	1 min.	30 min.
	Drill 2 places in corners		2 min.	
	Rough mill open area		5 min.	
	Finish mill open area		4 min.	
	Finish mill corners		10 min.	
	deburr		2 min.	
	Totals:	5 min.	39.5 min.	90 min

Assumptions:
All surfaces must be machined.
No residual stress.
No considerations are given for inspection.
All times a guesstimates.
Quantities are low enough that no high cost production line techniques are applied (i.e. rotary transfer, etc.).

135 minutes

Figure 56. Manual process plan from Engineer A and B for test case 3.

The variation in times and process plan shown in Figure 56 was typical across the test cases. Each manufacturing engineer has his/her way of doing things, their internal rules, and tends to stick with what they know. The AMFA software will allow the designer to have access to all of a company's available manufacturing processes as well as highlight standard corporate best practice process plans instead of individual personal preference. The speed up from using the corporate best practice as well as enhanced communication between designer and manufacturing will greatly reduce development time. In this case, the manual process creation time was approximately 30 minutes and the AMFA process time was nearly instant and provided a wide range of options. Based on the testing by the community, a conservative number of 5 minutes represents the time that is required for a designer to start AMFA, import a STEP file and post process the derived plans. AMFA enables a 6x speed improvement in creation of feasible process plans for desired geometry. Further benefit in quality and efficiency is gained by using corporate best practice which includes manufacturing cost reduction efforts as well as controlled corporate preferred manufacturing processes.

During the testing it was noted by the manufacturing engineers that the AMFA software pulls the designer into the manufacturing realm early on in the design process, which is seen as a major benefit from the manufacturing engineer's viewpoint.

The comparison between the manually created process plans and the AMFA process plans for test part 3 are shown in Figure 57.

	Manual process time range (min)	AMFA process time range (min)	Reasonable AMFA cost prediction? (Y or N	Do one of the plans match the manual plan? (Y or N)	Are redundent plans listed	Are sub optimum plans listed (Y or N)	DFM issues tab working (YES/NO)	Comments
								The results shown here are typical for the test cases. AMFA does tell the user when the geometry is not currently handled. The process plans created by AMFA match the manual plans and are all feasible. Some redundant plans are produced but ordered in different order. The geometric reasoning engine is producing Preliminary manufacturing process planning.
Test Case 3	135 - 170	140 - 195	Y	Y	Y	Y	Y	

Figure 57. Comparison sheet for test case 3.

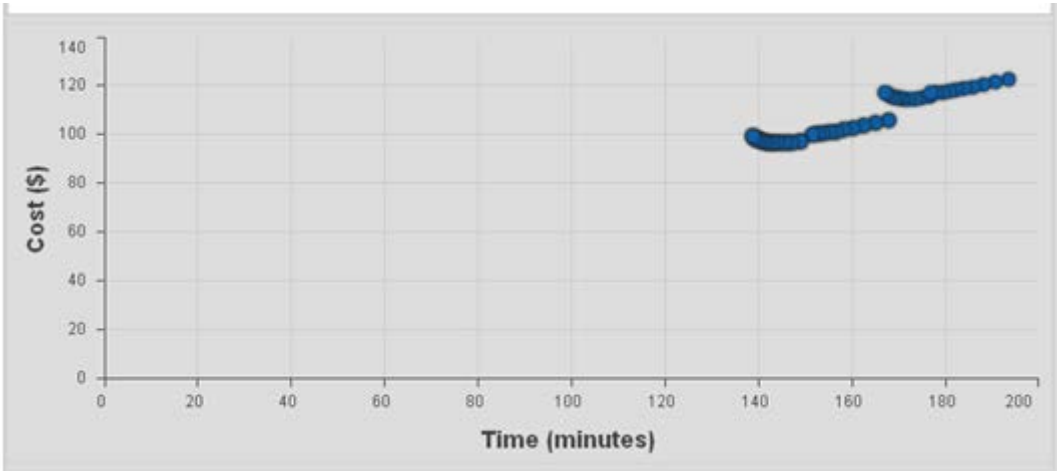


Figure 58. AMFA cost predictions for test case 3.

The testing results shown for test case 3 in Figure 57 and Figure 58 are typical for test case 1, 2, 4 and 6 so only test case 3 results will be documented in this report. Test case 5 was not processed because the sheet metal rules were turned off for the final phase of AMFA development and not explicitly required by contract.

The times reported in Columns 2 and 3 on Figure 57 for the manually created process plan and the AFMA create process plan do overlap so the AMFA process plan and is deemed feasible.

The cost reported in Column 4 on Figure 57 is deemed reasonable for a mid-size general purpose hardware manufacturer. The AMFA cost range can be seen in Figure 58.

Columns 4 to 8 for the six test cases, except for test case 5, were all scored as “Yes” so only test case 3 results will be shown. It should be noted that the AMFA software currently processes each solid model sub-partition as a unique machining operation when in reality the partitions that share a common face, tool and machine should be processed in one step. This is not deemed an issue because it should be easy to collect partitions based on the geometry and partition’s attributes and merge the partitions into one operation in a future release of AMFA.

The easy to follow layout of the AMFA software graphical user interface allowed the manufacturing engineer to visualize the merger of partition in the current version of the software so as to comment on the matches between the AMFA process plan and the manually created process plans.

The AMFA software technology demonstrator is deemed successful because AMFA software was able to produce feasible plans automatically, provided the designer with usable feedback through an easy to use graphical user interface in a real time environment (solutions returned to the designer in an hour) for all five test parts. Since AMFA provides numerical output from its manufacturing analysis it can be linked into current automated modular analysis processes that are tied to ”working” parametric solid models and used in industry today to robustly design corporations product. The AMFA software will bring manufacturing analysis into today’s concurrent engineering realm and further enhance industries performance to market.

5.0 CONCLUSIONS

We presented AMFA, an automated manufacturability feedback analysis tool - that automatically assesses whether a part can be manufactured as the designer specifies, or provides the designer feedback about what part of the design specification cannot be met given the constraints of a foundry. In addition, we described in detail the constituent pieces of innovations that collectively make up the AMFA tool.

AMFA's value stems from the role it fulfills in today's Product Lifecycle Management (PLM) tool chain – both for the DARPA AVM program and for the Industry. In today's PLM world, companies are deploying global PLM networks in which the solid model geometry of their products, sub-systems and components are linked with business and analysis data and can be shared with global team members, partners and customers in real time. These connected PLM databases and tools allow for the creation of designs that satisfy the customer's functional and performance requirements and support the creation of analysis data such as structural, thermal, dynamic and cost analyses.

The missing piece for a complete automation of the design and manufacturing process is the coupling between the design phase and the manufacturing phase. Today these coupling issues are typically resolved through multiple design-build-test-redesign iterations leading to longer schedules, capital-intensive manufacturing costs, reduced reliability and limited reconfiguration and reuse capability of the manufacturing enterprise. AMFA provides this missing concurrent engineering capability by bringing real world manufacturing constraints forward into concurrent design cycle, by identifying manufacturing constraints of a foundry, and thereby minimizing the time and cost of the manufacturing cycle.

BIBLIOGRAPHY

[1] "Software: Open CASCADE Technology, 3D modeling & numerical simulation," © OPEN CASCADE, 2000-2012. [Online]. Available: http://www.opencascade.org/.

[2] W. Fu, P. Radhakrishnan, A. A. Eftekharian, M. I. Campbell and C. Fritz, "A Graph Grammar Based Approach to Automated Manufacturing Planning," in *ASME International Design Engineering Technical Conferences IDETC*, Chicago, IL, USA, 2012 (in review).

[3] "Software: GraphSynth - Design, Implement, and Test Graph Grammars," [Online]. Available: http://www.me.utexas.edu/~adl/graphsynth/.

[4] M. I. Campbell, "GraphSynth2: software for generative grammars and creative search," 7 2 2012. [Online]. Available: http://graphsynth.com/.

[5] "Software: Fast Downward," [Online]. Available: http://www.fast-downward.org/HomePage.

[6] B. V. Blarigan, M. I. Campbell, A. Eftekharian and T. Kurtoglu, "Automated Estimation of Time and Cost For Determining Optimal Machining Plans," in *ASME International Design Engineering Technical Conferences IDETC*, Chicago, Illinois, USA, 2012.

[7] "Software: FeatureCAM, Product page," [Online]. Available: www.featurecam.com.

[8] J. R. Walker, Machining Fundamentals, Goodheart-Willcox, 2004.

[9] R. R. Kibbe, R. O. Meyer, J. E. Neely and W. T. White, Machine Tool Practices, Prentice Hall, 2009, p. 800.

[10] S. Kalpakjian and S. R. Schmid, Manufacturing Processes for Engineering Materials, Pearson Education, 2008, p. 1018.

[11] H. El-Hofy, Fundamentals of Machining Processes: Conventional and Nonconventional Processes, CRC/Taylor & Francis, 2006, p. 452.

[12] R. Enparantza, O. Revilla and A. Azkarate, "A Life Cycle Cost Calculation and Management System for Machine Tools," in *13th CIRP International Conference on Life Cycle Engineering*, 2006.

[13] "Software: HTML5," [Online]. Available: http://www.w3.org/.

[14] "Software: JavaScript," [Online]. Available:

http://www.oracle.com/technetwork/index.html.

[15] "Software: Sencha's ExtJS (Library)," [Online]. Available: http://www.sencha.com/products/extjs.

[16] "Software: Three.js WebGL (Library)," [Online]. Available: https://github.com/mrdoob/three.js/.

[17] M. Helmert, "The Fast Downward Planning System," *Journal of Artificial Intelligence Research*, vol. Volume 26, no. Issue 1, May 2006.

APPENDIX A: PROJECT TEAM

The PARC team consists of Palo Alto Research Center (PARC), University of Texas at Austin (UT Austin), Mission Critical Technologies (MCT), Arizona State University (ASU), and Rolls Royce LibertyWorks.

As the Prime, PARC expertise includes that of planning with preferences, reasoning about preference metrics, logical approximations for the derivation of relaxations and technology for symbolic reasoning (for keeping large search spaces tractable). For the iFAB program PARC used its state-of-the-art planning and knowledge representation expertise to reason about manufacturability results provided by the heuristic search for a process plan. The PI, Tolga Kurtoglu manages the Automation for Engineered Systems research area at PARC and is a Program Manager for the Software Integrated Systems program also at PARC. In addition to being PI on this iFAB effort, he was a co-PI on the PARC led META-II project "Formal Co-Verification of Correctness of Large Scale Cyber-Physical Systems During Design". The Co-PI of the iFAB effort was Dr. Christian Fritz, a research scientist at the PARC. PARC team members also included Dr. Eric Huang, Mr. Greg Burton and Dr. Saigopal Nelaturi.

University of Texas at Austin has a long history of research computational synthesis research including the automatic design of sheet metal components, multi-stable MEMS devices, function structures, and electro-mechanical configurations. Prof. Matthew Campbell led the iFAB research team at UT Austin which provided technology and expertise in graph-grammar based search for graph transformations. Specifically, the open-source GraphSynth software, developed by UT Austin, was exploited and extended to support the search over sequences of machine operations, in order to either find or refute the existence of a viable process plan. UT Austin team members also included Wentao Fu, Ata A. Eftekharian, and Benjamin Van Blarigan.

Mission Critical Technologies (MCT) provided experience in design automation and optimization, knowledge based decision support and knowledge representation of design for manufacturing, and computational design synthesis using graph grammars. The MCT effort was led by Yorgos Stylianos, Chief Science and Technology Officer, President & CEO at MCT.

Arizona State University's Design Automation Lab (DAL) has active research programs in Intelligent CAD systems, Concurrent Engineering, Design for Manufacturing, and Tolerance Modeling. The ASU effort was led by Dr. Jami Shah, director of DAL.

Rolls Royce LibertyWorks provided expertise in practical manufacturing and assembly for defense projects, hence ensuring realistic requirements, supporting testing and validation, and preparing for the transition of the developed representations and tools into production use. This includes the provision of realistic metrics and their assessment for specific designs. The Rolls Royce Corporation spans civil aerospace, defense aerospace, marine, energy, nuclear and services industries with a wide and diverse knowledge of engineering and manufacturing capabilities necessary to design and build innovative products, sub-systems and components for their global customer base. Mr. Joseph Rasche was the lead at Rolls Royce.

LIST OF ACRONYMS AND ABBREVIATIONS

ACRONYM	DESCRIPTION
3D	Three Dimensional
AMFA	Automated Manufacturability Feedback Analysis
ASU	Arizona State University
AVM	Adaptive Vehicle Make
B-rep	Boundary Representation
C2M2L	Component, Context, and Manufacturing Library
CAD	Computer-aided Design
CAM	Computer-aided manufacturing
CNC	Computer Numerical Control
CTF	Constraint Tolerance Feature format
DARPA	Defense Advanced Research Projects Agency
DFM	Design for Manufacturability
DRF	Datum Reference Frames
FANG	Fast Adaptable Next-Generation Ground Vehicle
FSM	Foundry-Style Manufacturing
GB	Giga Bytes
GD&T	Geometric Dimensioning & Tolerancing
GHz	Giga Hertz
GUI	Graphical User Interface
GXML	Graphic purpose Extensible Markup Language)
iFAB	Instant Foundry Adaptive through Bits
jpeg	Joint Photographic Experts Group
json	JavaScript Object Notation
LHS	Left-hand side
MFA	Manufacturability Feedback Analysis
MRAP	Mine Resistant Ambush Protected vehicles
OEM	Original Equipment Manufacturer
PARC	Palo Alto Research Center
PDDL	Planning Domain Definition Language
PLM	Product Lifecycle Management
RHS	Right-hand Side
STEP	Standard for the Exchange of Product model data
VMC	Vertical Machining Center
XC2V	Experimental Crowd-derived Combat Support Vehicle
XML	Extensible Markup Language