Proceedings of DETC01
2001 ASME Design Engineering Technical Conference
September 9-12, 2001, Pittsburgh, Pennsylvania, USA

DETC2001/DFM-21199

An interactive aid for designing and planning heterogeneous layered prototypes

Jonathan Clark, Liba Xia and Mark R. Cutkosky
Center for Design Research
Stanford University
Stanford, CA, 94305
clarkj@cdr.stanford.edu, http://www-cdr.stanford.edu/biomimetics

ABSTRACT

We describe a Java process editor to assist designers with the creation of complex, multi-material prototypes via a layered manufacturing process. Designers should generally participate in the manufacturing planning of such parts because many planning decisions are based on functional considerations. For maximum flexibility, the process planning tool should work in a collaborative environment with multiple CAD systems. These considerations have lead us to develop an interactive editor that is built atop a Java agent infrastructure and communicates with commercial CAD systems via their applications programming interfaces.

1. Introduction

Layered rapid prototyping appears to be splitting into two fields. The first, exemplified by commercial processes such as stereolithography, fuse deposition manufacturing, and selective laser sintering aim to realize the idea of a "3D laser printer" that can rapidly produce three dimensional prototypes from arbitrary CAD models. The second, exemplified by shape deposition manufacturing (SDM) [Merz, *et al.* 1994], focuses on functional prototypes that would be difficult to create with conventional manufacturing methods. Metals, ceramics and high-strength polymers are employed in varying combinations to achieve unique engineering properties [Fessler, *et al.* 1997; Weiss, *et al.* 1997]. Sensors, actuators and other components are embedded to create robust, integrated designs [Cham, *et al.* 1999]. Process planning is inevitably more complex with these processes and it is not possible to treat them as a black box that produces arbitrary parts from CAD models. For best results, the designer must participate in the manufacturing planning.

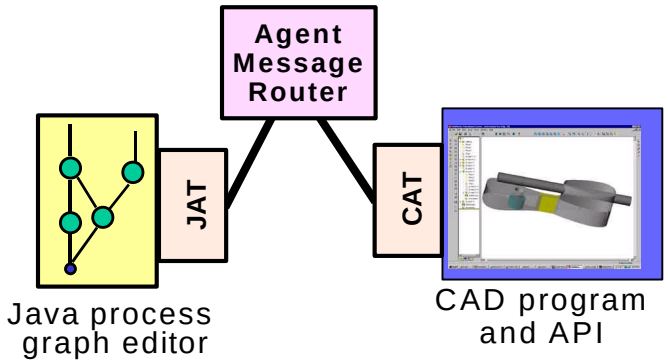

Figure 1. Graphical process editor communicates with a commercial CAD system to help the designer develop designs and plans for heterogeneous parts.

Unlike stereolithography, processes like shape deposition manufacturing currently remain confined to a few research laboratories. However, they are not necessarily difficult or expensive to undertake. For example, the only capital equipment needed to build high-strength polymer parts with embedded components, a combination of hard and soft regions and fiber reinforcement are: a computer, a CNC mill and a vacuum chamber. But without software for decomposing new designs into suitable sequences of geometries and process steps, it is difficult for designers to take advantage of the process. A goal of our work is therefore to create tools that will allow designers who are not experts in shape deposition manufacturing, but understand the functional and physical requirements of their designs, to guide the process planning so that they can obtain high quality parts.

In the following sections we briefly review the main considerations in process planning for shape deposition manufacture of heterogeneous parts and explain the difficulties associated with automating it. We then describe a semi-automated approach that allows designers to guide the decomposition process. The approach draws upon previous work on a "design by composition" algorithm [Binnard and Cutkosky 2000] in which designers compose new designs from libraries of previous designs and primitive elements that are stored along with partial process plans. We have implemented our approach with a Java agent that interacts with a commercial CAD system. Thus, the designer can develop her design within the CAD system and edit the corresponding process graph in a separate Java window. Changes in the graph are reflected in the CAD model and vice-versa. The approach is illustrated with the design of a robotic leg that includes embedded components and a compliant region connecting sections of structural materials. Functional considerations quickly reduce the number of possible process plans down to just a few.

2. Overview of Shape Deposition Manufacturing of Heterogeneous Parts

Shape Deposition Manufacturing (SDM) is a layered prototyping process that allows for the creation of heterogeneous parts. The concepts and agent software described in this paper have been developed with SDM in mind, but can be adapted to any layered manufacturing technique that includes multiple materials and embedded components.

SDM differs from many commercial layered processes in that there are both material addition and removal steps. After a material is added (e.g., through casting or laser fusing of metal power) it is shaped using a machine such as a CNC mill (Figure 2). The use of casting allows irregular geometry to be created. Machining provides precise finishes and close tolerances and reduces the variations in shape due to cooling or shrinking of the part material. Parts are built up by depositing and shaping alternating layers of part material and sacrificial support material which is later removed by melting or etching [Merz, *et al*. 1994].

The creation of precisely controlled internal surfaces with each shaping step facilitates the placement of embedded components such as bearings, sensors, microprocessors or reinforcing fibers. Figures 3a and 3b show a small robotic leg featuring several embedded components including a piston, two valves, a pressure sensor, and an amplifier circuit. By avoiding conventional assembly with mechanical fasteners, the part is both simpler to manufacture and more robust.

Different materials can also be deposited during each cycle producing heterogeneous parts with different regions having vastly different physical properties. Figure 3c shows a robot mechanism with regions of soft material that create a spatial linkage (similar to the front suspension linkage of an automobile) with built-in flexures instead of pin joints.

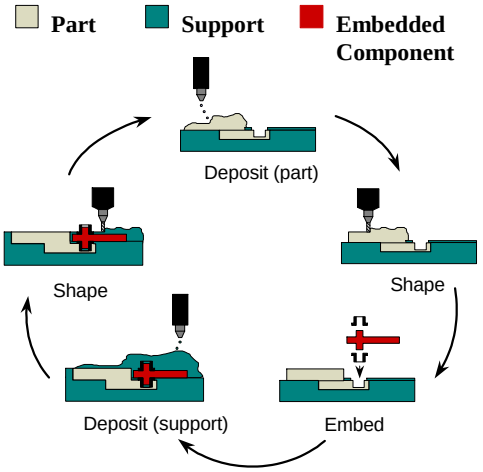

Figure 2. The SDM process cycle involves the addition and shaping of part and sacrifical support material. Components such as bearings or sensors can be added after any shaping step.

This flexibility in the use of materials and pre-fabricated components greatly increases the design space available to the designer. However, it also leads to complexity that requires the designer to participate in the planning process to ensure that all functional requirements are satisfied. For example, to achieve good fatigue life, the flexures in Figure 3c must not be machined on any surfaces that experience tensile strains.

3. Process Planning Issues

The fundamental process planning constraint with SDM is that the material removal process must have access to any surfaces that require shaping. If CNC machining is used for material removal, this means that the cutting tool must be able

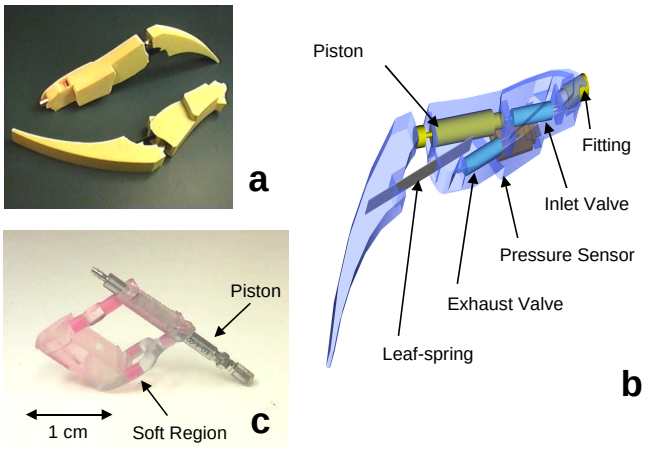

Figure 3. SDM mechanisms with embedded components and integrated hard and soft materials: (a) robot leg, (b) cut-away view of leg showing embedded components, (c) linkage with embedded piston and compliant flexures.

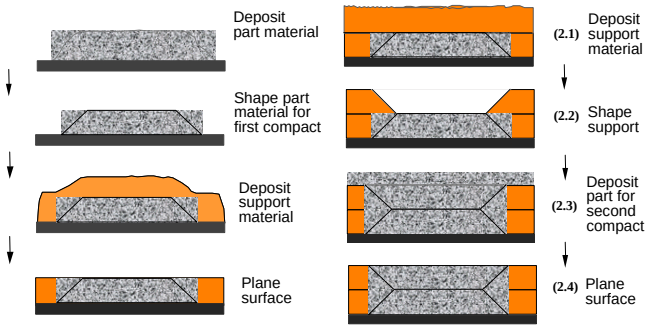

Figure 4. Downward-facing surfaces must be cast into a previously shaped mold of support material. Therefore, parts with undercuts and overhangs must be decomposed into multiple cycles. Here we see a part with two part material compacts created in two process cycles.

to access all surfaces. Any downward-facing surfaces on the part are shaped by casting them into previously machined support material. Thus, whenever a part has overhanging features it must be divided into two or more shapes that are created in separate cycles of the SDM process, as shown in Figure 4. A region of a part that can be fabricated in a single cycle of the SDM process is termed a "compact" in the literature [Merz, *et al.* 1994; Rajagopalan, *et al.* 2000]. Compacts may consist of part material, support material, or embedded components.

Once a build direction has been chosen, the decomposition of a part into compacts is the first step of process planning. Algorithms for automatic decomposition of arbitrary shapes into compacts have been discussed in the literature [Ramaswami 1997]. The result of decomposing a part into compacts is expressed in two graphs: the compact adjacency graph (CAG) and the compact precedence graph (CPG) [Pinilla, *et al.* 1997; Binnard and Cutkosky 2000] as shown in Figure 5.

The adjacency graph captures the connectivity among the compacts. The precedence graph is a directed acylic graph that represents the order in which compacts must be fabricated, starting at the bottom of the part and working upward along the

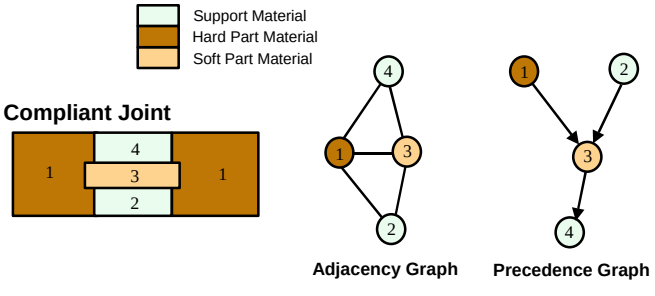

Figure 5. Compliant joint with corresponding compact adjacency graph (CAG) and precedence graph (CPG).

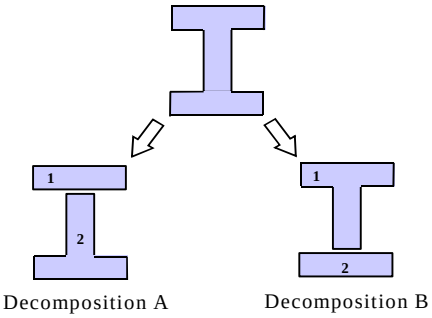

Figure 6. Two alternative decompositions for an I-beam structure.

build direction. In general, the CPG is only partially ordered. As Pinilla *et al.* [1997] have shown, there are efficiency advantages to deferring commitment to a particular total ordering as long as possible and ultimately until the part is undergoing fabrication in a manufacturing facility.

The decomposition of a part into compacts is not unique. For example, Decompositions A and B in Figure 6 represent two ways of making an I-shaped structure. Depending on whether greater strength is desired at the top or bottom of the vertical web of the I beam, the left or right alternative may be preferred. This is an example of a functional consideration leading to a choice in process planning. Typically, some decompositions will produce unmanufacturable parts (e.g. parts requiring cutting tools of extremely small diameter or that present difficulties when trying to embed components). Additional considerations are often encountered when working with combinations of different part materials and embedded components [Cham, *et al.* 1999].

Figure 5 illustrates a common problem. The objective is to create a structure with two regions of hard material joined by a flexible element of soft material. The cross section on the left shows a simple decomposition into 4 compacts, along with the associated CAG and CPG. For good bond strength and fatigue life the soft material must be cast into freshly machined hard material. Therefore compact 1 must precede compact 3. (Note that compact 1 does not have to represent a singly connected

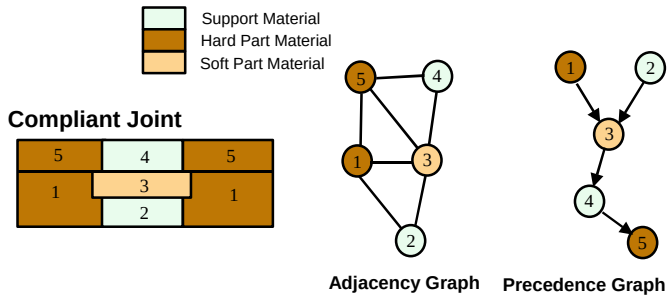

Figure 7. Revision of design in Fig. 5 for manufacturability and to achieve better surface finish on compliant joint.

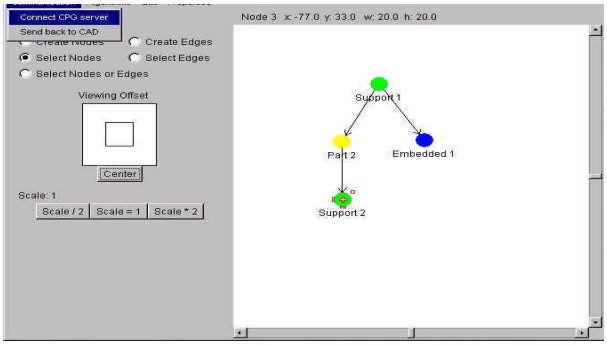

Figure 8. View of Java process editor and compact precedence graph (CPG).

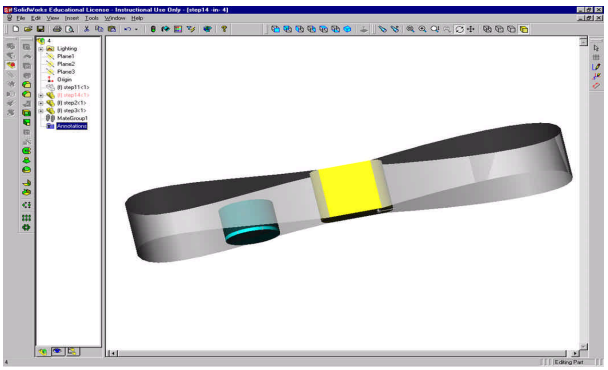

Figure 9. CAD model of a robot leg assembly in process, corresponding to the graph window in Figure 8. (Support material is not displayed in this view.)

region in space; it must only represent a region that can be created in a single SDM cycle.) The soft material must also be supported; therefore compact 2 must precede compact 3. However, to obtain a good finish and flatness, the upper surface of the soft material should be shaped by a material removal process. General machining operations are not possible on the soft material, but planing is possible. The solution is to split compact 1 into a lower and upper part. The resulting CAG and CPG are shown in Figure 7.

Manufacturing considerations such as these led Binnard and Cutkosky [2000] to develop an algorithm in which designers compose new designs from libraries of previous designs and primitive elements. Each member of the design library is stored as a set of compacts of part material encased in compacts of support material, along with a CPG for the set. When a designer merges a library element into an evolving design, the element's CPG is merged with the CPG of the design. Extensions to the basic algorithm address the merging of library elements with embedded components [Cham, *et al.* 1999]. A C++ software implementation of the basic algorithm was created as a "plug-in" for the AutoCad[1] design software.[2] An advantage of the design-by-composition approach is that a particular decomposition into compacts, and the associated ordering of process cycles, can be enforced at the time the library element is created and stored.

Although Binnard's algorithm is provably correct, it runs into some difficulties when implemented with commercial CAD software. The first problem is that when designers are working on novel designs there are few library elements to build from. Creating new shapes entirely from primitives like cylinders and cubes is cumbersome and requires many Boolean operations among different part and support material geometries. The Boolean operations are typically slow compared to surface operations in current CAD systems and more prone to errors.

1. AutoCADARX14, from Autodesk Inc.
2. The plug-in software can be downloaded from http://www-cdr.stanford.edu/interface/software.html

Afterward, many compacts can be merged to simplify the design and process plan, but this step requires human input because it depends on functional considerations like those discussed for Figures 5-7. Until these issues are resolved, we have developed a simpler, less automated approach when working on new designs. In this approach the designer controls the process plan formation through a separate graph editor window. This approach follows a trend in other manufacturing fields such as injection-mould design and printed circuit board assembly in combining expert systems with a platform independent 2-way process planning/design tool [Mok, et al. 2001][Liau, et al. 1995].

4. A Java process planning editor and agent interface

There are several possible approaches to help designers participate in planning a specialized manufacturing process. One approach is to provide a CAD interface that is customized for the manufacturing process. A good example of this approach is the Cybercut interface at U.C. Berkeley [Brown and Wright 1998]. Another possibility to is provide a plug-in utility that works with a commercial CAD program, as was done by Binnard and Cutkosky [2000]. However, for maximum flexibility in a collaborative environment, the process planning tool should not be tied to a particular CAD program or interface.

The following philosophy guided our development of an interactive process editor:

- The editor should be kept relatively independent of the CAD program, since the graph operations do not depend on the details of the solids operations.
- The editor should be interactive and should communicate directly with the CAD program so that process plan modifications are reflected in the CAD system and changes to the CAD model are reflected in the graph display of the process plan.
- The editor should be platform-independent, so that it works with programs on Unix and Windows environments.
- The editor/CAD program communication should be robust. Messages should be queued and buffered so that if either

program crashes it can be restarted and the dialogue can easily be brought back to the previous state.

Agent messages	consequences
CAD interface agent (CAD)	
register	Register and establish communication with message router
load design	Load file containing solid models or part in terms of compacts and precedence information
new compacts (list)	PG: create nodes
delete compact	PG: delete node
merge compacts (list)	PG: merge nodes
split compact	PG: create node
select compact	Highlight compact PG: select node
Process graph agent (PG)	
register	Register and establish communication with message router
merge nodes (list)	CAD: merge compacts (list)
create link (from, to)	Create precedence constraint between corresponding compacts
delete link	Delete corresponding precedence constraint.
select node	Highlight node CAD: select compact
simulate	Traverse graph. CAD: display intermediate geometry (union of compacts) at each step.

Table 1: Operations and communications between editor agent and CAD program plug-in

These considerations have lead us to develop a graph editor agent that uses the Java Agent Template (JAT*Lite*) package for communications, shown in Figure 8. The corresponding CAD model is shown in Figure 9. JAT*Lite* [Jeon *et al.* 2000] is an open-source[3] Java package developed in previous work at our laboratory for creating agents that communicate over the Internet. At the lowest level, communications are based upon TCP/IP. JAT*Lite* agents communicate through an agent message router (AMR) that provides services for login, registration, and message queueing (similar to an email server). The graphical interface is built on the *Visualizing Graphics with Java* (VGJ) graph package from Auburn University [McCreary

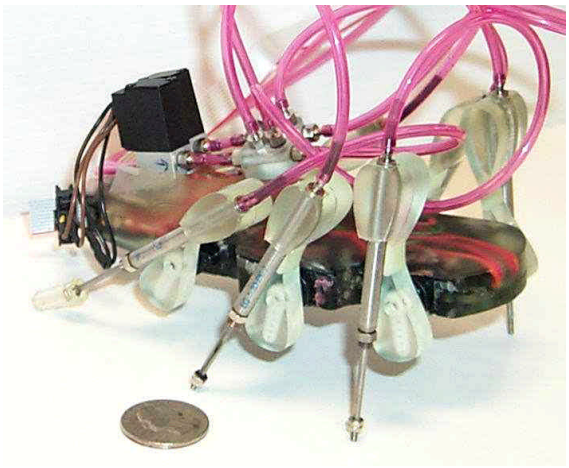

Figure 10. "Sprawlita," a small (0.25 Kg) shape deposition manufactured robot with embedded components and multiple materials.

1998].[4] A C++ counterpart to the Java Agent Template called CAT has been developed for use with C/C++ programs. The CAT interface is used with the applications programming interface of a CAD program to provide communication with the process editor agent. The basic communications structure is shown in Figure 1. The messages that can be exchanged between the process editor and the CAD program are summarized in Table I. With this system in place, changes in either the CAD system or the graph window can be synchronized. To give a better feel for how the editor is used for designing new heterogeneous parts, we present a detailed example in the following section.

5. Example: design of a compliant robot leg

Figure 10 shows the "Sprawlita" hexapedal running platform developed at Stanford. More information about the biomimetic design and performance of the robot can be found in [Clark, *et al.* 2001; Cham, *et al.* 2000].

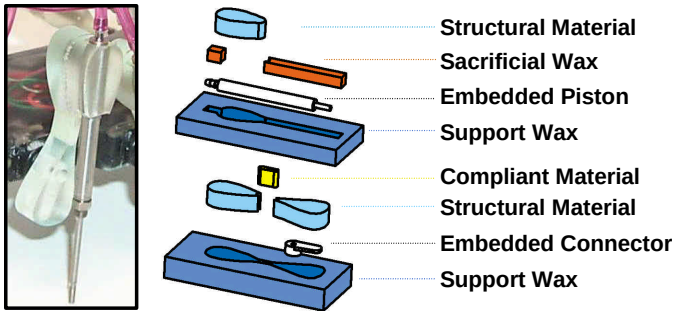

Figure 11. Leg detail from Sprawlita, showing embedded piston and compliant flexure.

3. JATLite is available under the GNU public license at http://java.stanford.edu

4. http://www.eng.auburn.edu/csse/research/ research_groups/ graph_drawing/graph_drawing.html

The design and construction of its compliant legs (Figure 11) provide a good example of the issues and approach discussed in this paper. The design is simple enough to show the entire process and illustrates the role of functional considerations in guiding the process planning.

Design considerations

The legs for the robot should be small, lightweight and robust. In addition they should be have connectors for attaching to servo motors on the robot body and pistons for generating thrust. The legs should also include a region of soft material to provide a controlled amount of rotational compliance and damping in the sagittal plane of the robot. These considerations make SDM the preferred manufacturing method. To fabricate small legs with these features using conventional assembly methods would be tedious and expensive. The SDM legs are also more robust because they have no tiny fasteners to fail or work loose. These and other advantages of compliant mechanisms have been articulated by other researchers [Ananthasuresh, 95].

Process sequence

The following steps outline the process that the designer goes through designing the leg while also developing the process plan for creating it.

Setup: The designer wants to make a robot leg primarily of hard urethane, with a soft flexure at the center. In addition, she would like to embed a piston and a connector for attachment to a servo motor. Given the relatively flat shape of the leg, there are two obvious choices for build direction. After some thought, the designer picks the direction of Figure 11 so that the piston can be added toward the end of the manufacturing process. The designer launches the CAD software and loads the process planning and communication plug-in. The plug in provides an extra pull-down menu to support the operations in Table 1.

Step 1: The designer begins by placing a block of support material into the CAD workspace. This becomes compact #1. If the designer clicks “submit” in the plug-in menu, the corresponding first node in the graph will appear in the graph editor window.[5]

Step 2: The next step is to create an outline for the leg and extrude it in the growth direction. The resulting dog-bone shaped solid becomes the first compact of part material and corresponds to the second node in the process graph.

Step 3: The part material compact is subtracted from the support material to form a pocket that will be used for casting the first layer of the leg. The modified support material and the leg material now form a two-part assembly in the CAD system (Figure 12).

5. The “submit” command can be issued at any time and the Java editor will display all new nodes since the previous submission.

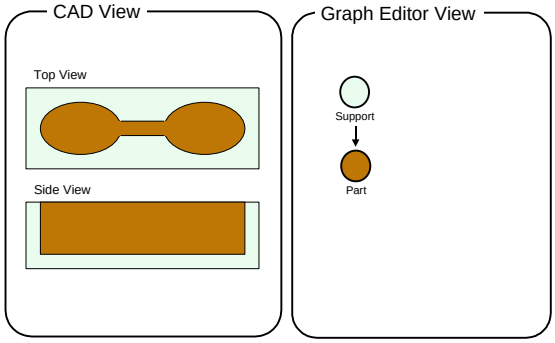

Figure 12. Design and graph at end of step 3: part material and shaped support material form a simple assembly in the CAD window and a two-node graph in the process window

Step 4: The user now loads and positions a previous design

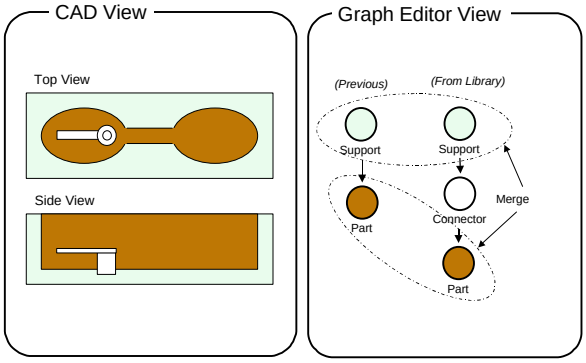

Figure 13. Assembly of compacts and process graph after adding an embedded connector in Step 4. The connector graph can be merged with the graph for the leg.

(perhaps from a small design library) for “embedded connector” (or perhaps simply “embedded vertical cylinder”). This design consists of compacts of support material, embedded component, and encapsulating material and accompanying precedence relationships (support ↦embedded component ↦encapsulating material). Loading this design and clicking the “submit” button results in a new sequence of nodes that appears in the graph editor, as shown in Figure 13. At this point, the designer notes that the beginning and end nodes of the two chains can be merged because they are of the same material type. Thus, when the user merges the nodes in the graphical editor the support compact for the connector is automatically merged (unioned) with the support material compact for the leg and the part material that encapsulates the connector is unioned with that of the leg itself. It is up to the designer to determine whether the resulting unions represent valid compacts. (As discussed in Section 6, geometric validity tests are a subject of future work.) Note also that the connector extends outside the leg. Therefore, the geometry of the base compact of support material is modified by subtracting the connector from it. This creates a circular depression in the bottom of the pocket for locating the connector when it is embedded.

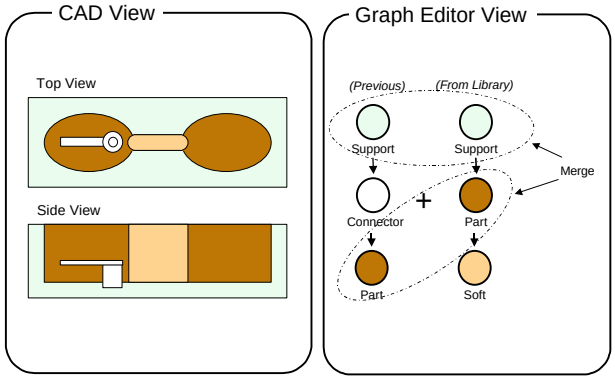

Figure 14. CAD model and process graph after adding a compliant flexure in Step 5.

Step 5: The addition of the flexure is similar to the previous step. However, for best fatigue life, the flexure material should be cast directly into freshly machined hard part material. In addition, the soft material should not undergo any machining on surfaces that will experience tensile stresses. The process plan fragment for a flexure is (support ↦hard material ↦soft material). Figure 14 shows the CAD window and process graph window after the flexure is loaded. Again, the designer merges compacts so that the flexure is cast into a pocket cut directly into the leg material. A screen shot of the CAD model is shown in Figure 9. It is an assembly of four parts: support material, connector, hard leg material and flexure. (The support material is not shown for clarity.)

Step 6: At this point the designer realizes that she wants a through-hole to the connector, so she creates a hole by subtracting it from the leg material compact. The hole will be drilled into the leg material, so it must follow the part material node in the graph. However, there is no ordering constraint between the hole and the flexure, so a branch is created in the process graph, as shown in Figure 15.

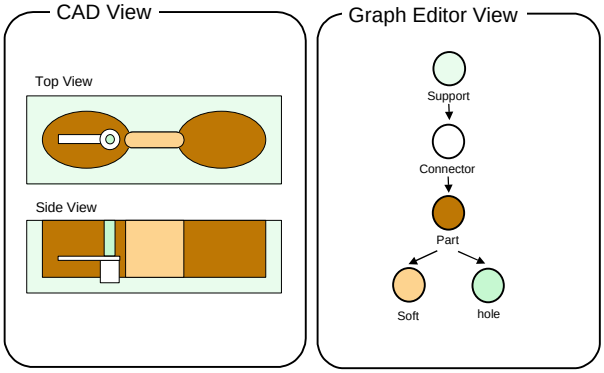

Figure 15. In Step 6 a hole is added to the part. This step is independent of the soft flexure.

Step 7: Next the designer adds the piston. Like the connector, the piston is an embedded component that is typically located by machining some support material, assembling the piston in space and then casting part material around it. The piston blocks access to the flexure and through hole and therefore it must be added after they have been created (i.e., the compact of part material that encapsulates the piston cannot be merged with the first compact of hard material). The piston process plan is therefore manually appended to the existing plan as shown in Figure 16.

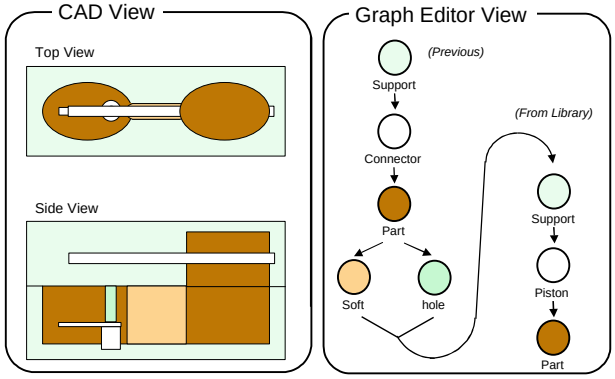

Figure 16. An encapsulated piston is added to the leg and its process plan fragment is appended to the previous plan.

The design and high-level process plan are now complete. The graph is then traversed creating a series of geometries that can be loaded into a CNC toolpath planning program to create the part. The plan involves three casting cycles: two for hard material and one for soft. Each casting step is followed by vacuum degassing and a cure time of 4-10 hours before the next material removal step. Thus there is a strong motivation to merge compacts where possible, as in Steps 4 and 5 (Figures 13 and 14). This example demonstrates how having an interface which features: (i) the process graph and CAD model simultaneously visible, (ii) constraints enforced, and (iii) automatic propagation of changes from one view to the other helps the designer to quickly converge on a practical and efficient design and corresponding manufacturing process plan.

6. Conclusions and Future Work

The creation of heterogeneous parts with multiple materials and embedded components is necessarily more complex than the creation of homogeneous layered prototypes. Although efforts are underway to automate the decomposition and process planning of such parts [Ramaswami, 1996; Kao 1999; Rajagopalan, *et al.* 2000], the current state of the art is that designers must control the process planning to ensure that their requirements are satisfied. This situations is similar to the current state of the art in other fields involving combinations of geometric reasoning and functional considerations, such as injection moulding design [Mok, 2001], automated PCB assembly [Liau, et al. 1995] and automated fixturing [Zhuang,

1996]. An alternative approach in which designs are composed from library elements is promising [Binnard and Cutkosky 2000]. In this way, the creation of complex mechatronic parts may eventually approach the efficiencies seen in VLSI design and fabrication. But at present, this approach is also out of reach for complex heterogeneous parts. The designs are not yet standardized enough to facilitate the creation and use of libraries. Also, design by composition involves numerous Boolean operations on assemblies of multiple materials, which severely taxes the capabilities of today's desktop CAD systems.

In light of these difficulties, we have developed an interactive tool that helps the designer create process plans in parallel with an evolving design. The tool consists of a Java graph editor that communicates with commercial CAD software via a previously developed agent interface and communication language. This approach provides maximum flexibility, as it is easily adapted to work with any CAD system that provides Boolean operations, assemblies of parts and an applications programming interface. We demonstrated the use of this tool for designing and fabricating a robot leg with a compliant flexure and two embedded components. The CAD system used for this example was Solid Works,[6] but we have also done some experimentation with Solid Edge[7] and AutoCAD[8].

Several straightforward extensions are planned. Foremost among these is to implement a test for compactness (manufacturability) so that the user is notified of any invalid compacts that require splitting. This can be accomplished by passing a vertical line through each downward-facing surface in the compact. If the line exits and reenters the compact, the compact is invalid. Similarly, the process editor agent should detect any inadvertent cycles in the process graph.

Better integration with the second phase of process planning, cutter path generation, is also desirable. At present, the sequence of geometries that result from traversing the process graph are saved into files and exported to another program for toolpath planning. Given that toolpath planning can also involve difficulties which may require a revision to the process plan, it is desirable to be able to perform the toolpath planning step-by-step, for each process cycle, as the design evolves.

7. Acknowledgments

The authors thank the members of the Stanford DML and RPL teams for their contribution to the work documented in this paper. This work is supported by the National Science Foundation under grant MIP9617994 and by the Office of Naval Research under N00014-98-1-0699.

6. Solid Works 99, Solid Works Corp.
7. Unigraphics Solutions Inc.
8. Autodesk Inc.

8. References

Ananthasuresh G.K., Kota S. "Designing Compliant Mechanisms" *Mechanical Engineering*, Vol. 117, No. 11,November 1995, pp. 93.

Binnard, M., Cutkosky, M.R., "Building Block Design for Layered Shape Manufacturing," *Proceedings of the ASME Design Engineering Technical Conference, Atlanta, GA, September 13-16*, 1998.

Binnard, M., Cutkosky, M.R.,"A Design by Composition Approach for Layered Manufacturing," *ASME Transactions, Journal of Mechanical Design*, Vol. 122, No. 1, March 2000, pp. 91-101

Brown, S.M. and Wright, P.K., "A Progress Report on the Manufacturing Analysis Service, an Internet-Based Reference Tool," 1998, *Journal of Manufacturing Systems*, Vol. 17, No. 5, 1998.

Cham, J. G., Bailey, S. A., and Cutkosky, M. R., "Robust Dynamic Locomotion Through Feedforward-Preflex Interaction," *ASME IMECE Proceedings, Orlando, Florida, November 5-10*, 2000.

Cham, J.G.; Pruitt, B.L.; Cutkosky, M.R.; Binnard, M.; Weiss, L.E.; Neplotnik, G., 1999, "Layered manufacturing with embedded components: process planning issues," *ASME Proceedings, DETC '99.* Las Vegas, Nevada. September 12-15.

Clark, J. E., Cham, J. G., Bailey, S. A., Froehlich, E. M., Full, R. J., and Cutkosky, M. R., "Biomimetic Design and Fabrication of a Hexapedal Running Robot," *IEEE International Conference on Robotics and Automation, Seoul, Korea,May 21-26*,2001.

Fessler J.; Nickel, A.; Link, G.; Prinz, F.; Fussel, P., 1997, "Functional gradient metallic prototypes through shape deposition manufacturing," *Proceedings of the Solid Freeform Fabrication Symposium*, Austin, TZ, September, 1997, pp. 521-528.

Jeon, H., Petrie, C. and Cutkosky, M.R., "JATLite: A Java Agent Infrastructure with Message Routing" *IEEE Internet Computing*, Vol. 4, No. 2, Mar/Apr 2000.

Kao, J-H., "Process planning for additive/subtractive solid freeform fabrication using medial axis transform," Ph.D. Thesis, Stanford University, June 1999.

Liau, J., Young, R.E., and O'Grady, P., "Combining Process Planning and Concurrent Engineering to Support Printed Circuit Board Assembly," *Computers Ind. Engng* Vol. 28, No. 3, 1995.

McCreary, C., "Visualising Graphs with Java" a graph drawing and layout tool distributed under GNU public license at http:// www.eng.auburn.edu/csse/research/ research_groups/ graph_drawing/graph_drawing.html

Merz, R., Prinz, F.B., Ramaswami, K., Terk, M., Weiss, L.,

"Shape Deposition Manufacturing," Proceedings of the Solid Freeform Fabrication Symposium, University of Texas at Austin, August 8-10, 1994.

Mok, C.K., Chin K.S., and Ho John K.L., “An Interactive Knowledge-Based CAD System for Mould Design in Injection moulding Processes,” *Int J Adv Manuf Technol*, 17:27-38, 2001.

Pinilla, J. M., Kao, J. H., Binnard, M., Prinz, F. B., “The Compact Graph Format: an Interchange Standard for Solid Freeform Fabrication,” NIST Measurement and Standards Issues in Rapid Prototyping Workshop, Gaithersburg, MD, 1997.

Rajagopalan, S., Goldman, R., Shin, K. H., Kumar, V., Cutkosky, M. R., and Dutta, D., “Representation of Heterogeneous Objects during Design, Processing and Freeform Fabrication ,” *Materials and Design*, v. 22, no 3, pp. 185-197, December 2000.

Ramaswami, K., "Process Planning for Shape Deposition Manufacturing," PhD Dissertation, Stanford University, Stanford, California, 1996.

Weiss, L.E., Merz, R., Prinz, F.B., Neplotnik, G., Padmanabhan, P., Schultz, L., Ramaswami, K., "Shape Deposition Manufacturing of Heterogeneous Structures," *SME Journal of Manufacturing Systems*, Vol. 16, No. 4, 1997.

Zhuang Y, Goldberg K., “On the Existence of Modular Fixtures,” *International Journal of Robotics Research*. 15(5), December, 1996.