AFRL-HE-WP-TR-2004-0156

United States Air Force
Research Laboratory

Service Manual Generation (SMG)

Jeanette Bruno
Steve Linthicum

GE Corporate Research and Development
One River Road
Schenectady, NY 12301

Jeffrey L. Wampler

Air Force Research Laboratory

April 2004

Final Report for the Period April 2001 to April 2004

Human Effectiveness Directorate
Warfighter Interface Division
Cognitive Systems Branch
2698 G Street
Wright-Patterson AFB OH 45433-7604

NOTICES

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Please do not request copies of this report from the Air Force Research Laboratory. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield, Virginia 22161

Federal Government agencies and their contractors registered with the Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
8725 John J. Kingman Road, Suite 0944
Ft. Belvoir, Virginia 22060-6218

TECHNICAL REVIEW AND APPROVAL

AFRL-HE-WP-TR-2004-0156

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

//Signed//

MARIS M. VIKMANIS
Chief, Warfighter Interface Division
Air Force Research Laboratory

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
April 2004	Final	April 2001 - April 2004

4. TITLE AND SUBTITLE
Service Manual Generation (SMG)

5a. CONTRACT NUMBER
F33615-01-2-6000

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62202F

6. AUTHOR(S)
Jeanette Bruno, Steve Linthicum, Jeffrey L. Wampler

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
1710D109

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

GE Corporate Research and Development
One River Road
Schenectady, NY 12301

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory
Human Effectiveness Directorate
Warfighter Interface Division
Cognitive Systems Branch
Wright-Patterson AFB OH 45433-7604

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-HE-WP-TR-2004-0156

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The objective of this research was to enable the automatic generation of verified maintenance instructions. The approach to the problem was to divide the overall program into three main research areas. The first area was to develop state-of-the-art Computer Aided Design (CAD) analysis algorithms to generate valid assembly and disassembly sequences from the 3D CAD geometry. The second area targeted natural language techniques to generate understandable maintenance instructions, and the third area focused on algorithms that could support verifying the instructions in an immersive environment with force feedback. The program began with the development of a high level concept of the solution to each research area and a defined set of tasks for each research area that coincided with the high level concepts. As the research progressed and more was leaned about the problems and the true business needs, the concepts and subtask efforts were refined. This report documents, for each research area: 1) the solution approach, 2) the subtasks associated with each area, and 3) a discussion of how these subtasks add up to a solution to the original problem. For each subtask, a description of the results achieved (including algorithmic descriptions, etc.) and how lessons learned during the program affected each subtask. Following the research area descriptions is documentation for wrapping the three technology results into one comprehensive demonstration platform. The report then describes how this platform was used to demonstrate the technology capabilities to the user and the results of these demonstrations. The final phase of the report discusses the strengths/weaknesses of the technology, where it appears to be going and growth areas for the research.

15. SUBJECT TERMS
Maintenance Manuals, Computer Aided Design (CAD), Technical Manual Development, Algorithms

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR	54	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

1.0 Executive Summary

The vision of this program was to enable the automatic generation of verified maintenance instructions. Our approach to the problem was to divide the overall program into three main research areas. At the beginning of this program, we already had developed a high level concept of the solution to each area and defined a set of tasks for each research area that coincided with these high level concepts. As we progressed through the program and learned more about the problem, and the true business needs we refined these concepts and subtask efforts. In the following pages we document, for each research area, the solution approach we took, the subtasks associated with each area and a discussion of how these subtasks add up to a solution to the original problem. For each subtask we describe the results achieved under each subtask (including algorithmic descriptions, etc.) and how lessons learned during the program affected each subtask.

Following the research area descriptions we document how we wrapped the 3 technology results into one comprehensive demonstration platform. We describe how we used this platform to demonstrate the technology capabilities to the user and the results of these demonstrations. The final parts of the document discuss the strengths/weaknesses of the technology, where it appears to be going and growth areas for the research

2.0 Introduction

2.1 The Vision. The vision of this program was to enable the automatic generation of verified maintenance instructions. The idea was to develop advanced techniques for the analysis of 3D CAD geometry. These techniques would support automated maintenance procedure generation, converting the CAD-based steps into an understandable maintenance instruction and verifying the instruction material with a writer or technician in a virtual environment, complete with force feedback for human actions, simulating the actual performance of the task on a real product.

2.2 Overview. To achieve this vision we divided the program into 3 research areas. The first area was to develop state-of-the art CAD analysis algorithms to generate valid assembly and disassembly sequences from the 3D CAD geometry. The second area targeted natural language techniques to generate understandable maintenance instructions, and the third area focused on algorithms that could support verifying the instructions in an immersive environment with force feedback.

At first each research area focused on discovering techniques to overcome their technical hurdles: the sequence generation analysis was faced with very large data sets and search spaces, the natural language area needed a useful characterizations of the 'language' in the technical manuals and mechanisms to bridge CAD data and these characterizations, and the research into the immersive environment faced the challenge of being able to do real-time collision analysis and force feed-back computations on very large data sets. We analyzed each research area, and for each, determined an approach to developing a solution. These approaches are documented as a series of subtasks for each area. The pages that follow contain an overview of the approach for each research area and a detailed description of the subtasks in each area and the results of the work performed on each subtask.

As the research progressed and solutions to the main technical hurdles were developed, we also refined our understanding of each area to include an understanding of the additional requirements imposed by integrating the three areas with each other and with the overall aircraft engine design, build, deploy process.

From the initial vision of the program, it was assumed that the first targeted users of the technology would be the people who produced the engines' technical publications. As we gained a better understanding of what the technology could achieve and how the general large product manufacturing businesses operated, we came to understand that we needed to expand the scope of our user set. We realized that though the technical publications group developed detailed documentation describing maintenance procedures, they did not develop the procedures all by themselves. In reality, the maintenance procedures evolved (in an informal manner) over the course of developing the product. The following paragraphs describe this process and how it impacted the goals of the research.

The first step toward developing maintenance procedures: During the initial design phase, maintainability analysts assess the complexity of maintenance procedures for a proposed design. The formal goal of this analysis is to feed the results of the analysis back to the designers to help them assess the different design

options. Informally, this analysis also lays the groundwork for the part maintenance procedures. Though these analysts don't develop fully described maintenance procedures, they do determine the critical sequence of events necessary for a given maintenance procedure. That is, they don't determine the sequence for loosening or tightening fasteners (bolts, clamps, etc.) or using consumables, but they do determine which large parts are blocking each other and the order for removing/installing the large parts to achieve the desired maintenance goal.

The next step toward developing maintenance procedures: We also discovered that these procedures are expanded and become more fully described when the prototype engines are built. As part of the build up of the prototype engines, detailed assembly instructions are developed. Granted these instructions are not formatted as they would be in a published technical manual, and they do not need to meet external publication standards, but they do contain the step-by-step details of putting an engine together. These less formal assembly instructions are delivered to the assembly floor for the prototype engine build up. The actual build up of the prototype engine has the side effect of verifying the validity of the instructions. This does not imply that the business did not need to use the virtual environment to verify the instructions. On the contrary, we were informed that finding errors in the instructions during the prototype build process can be a costly event, and that it would very desirable to be able to use the virtual environment before building the prototype, to find and correct errors before the parts are sitting on the assembly floor.

Producing the technical publications: Finally, when the detailed product assembly, disassembly and maintenance instructions are produced, the technical publications experts use notes from the test build instructions to develop the formally published instructions. Typically, a complete overhaul procedure will be very similar to the test build up procedures and the smaller maintenance procedures tend to be subsets of the build up procedures. As such the procedures developed for the test build engines form the basis for the final production assembly, and maintenance procedures. Even if the final sequence is altered dramatically, the build up specifications such as torques, consumables, warnings, etc. need to be carried into the final publications.

Once we understood the process described above we realized that the technology being developed via this program should touch a much larger user community than just the technical publications group. We realized that to maximize its effectiveness, the technology should be used as follows. During the initial design phase, the maintainability analysts would use the virtual environment to discover and validate high-level maintenance procedures. These procedures could be captured and passed on to the prototype assembly planning, and detailed assembly instructions would then be developed and verified for the prototype builds. As such all three technologies, sequence generation, natural language generation and immersive reality would be used in the development of the prototype build procedures. Once the prototypes build stage is complete, production assembly planning could use the information from the prototype builds and again use the three technologies to adapt the build plans to meet the restrictions of the production assembly environment. Finally, the technical publications group would take the detailed, validated assembly procedures and high level (validated) maintenance procedures and modify them to clarify the presentation (note: they typically would not modify the actual procedure steps). The technical publications group could also use the virtual environment to validate the procedures, but it is much more desirable to find any problems early in the process (prior to tech pubs involvement) and thus reduce the cost of the errors.

We also realized that the virtual environment has a strong potential in the training and marketing arena.

The software developed under this program was a mix of C++, tcl/tk and java. For the most part the CAD analysis software used the public domain Visualization Toolkit (VTK) library [1]. The new CAD analysis algorithms that were developed under this program were implemented as an add-on library that sits on top of the VTK base. The user interfaces were written in Active Tcl/Tk [2]. The software developed under the natural language area was written in Java and used MYSQL [3] to implement the underlying relational database.

3.0 Technical Discussion

3.1 The Three Research Areas. As stated in the vision, the main idea behind this program was to develop advanced techniques for the analysis of 3D CAD geometry to 1. support automated maintenance procedure generation, 2. to convert the CAD-based steps into an understandable maintenance instruction, and 3. to

verify the instruction material with a writer or technician in a virtual environment, complete with force feedback for human actions, simulating the actual performance of the task on a real product.

This idea can be naturally divided into 3 functional areas:

- the ability to generate valid assembly and disassembly sequences from the 3D CAD geometry (Sequence Generation) ,
- the ability to generate understandable maintenance instructions from CAD based sequences (Task Generation), and
- the ability to use an immersive environment with force feedback to validate the instructions (Virtual Validation).

(As mentioned in the *Objective* discussion, as the program progressed, we discovered that the immersive environment would be used much earlier in the aircraft engine design process than originally anticipated, and for a slightly different purpose; to develop valid part removal information rather than to validate part removal information, but these new uses did not change the underlying technical challenge.)

The following pages describe the results of our efforts in each of the three areas.

3.2 Sequence Generation

Overview. Our approach to generating valid assembly and disassembly sequences from the 3D CAD geometry consisted of developing algorithms to generate disassembly sequences and then an algorithm to reverse a disassembly sequence to generate an assembly sequence. The problem boils downs to a solution search such that the solution defines a parts ordering and motion description that describes collision free motion for each part when the parts are moved in the selected direction(s) in the selected order. The difficult part of this problem comes from the fact that the search space is infinite: the part ordering aspect is only an $O(n^2)$ search space (for n parts) but the part motion concept represents a continuous, thus infinite search space. In order to make this a tractable problem, we needed to develop methods to reduce the search space to a computable size

Our basic approach was to limit the part motion analysis to discrete directions and discover pairwise part blocking/interaction information. (Our analysis would characterize how each pair of parts blocked each other's movements.) We then analyzed this part pair blocking information to discover a valid ordering of part motions such that each part would be removed in some (algorithmically) selected direction only after all parts blocking it in that direction had been removed.

The pair-wise part blocking information in essence describes, for each ordered part pair, (*partA, partB*), the directions that *partB* blocks *partA*. The first step toward making this a computable problem was to reduce the continuous directional search space. As such we introduced a discretized spherical space represented by a parameterized set of directional vectors, thus reducing the directional solution space to m discrete directions and reducing the overall solution space to an $O(mn^2)$.

The remaining computationally challenging aspect of the algorithm was in computing whether one part blocks another is a specified direction. The CAD geometry we chose to work with uses a polygonal approximation of the part. As such determining if one part blocks another in a given direction requires (for the worst case) comparing every polygon in the first part to every polygon in the second part. This aspect of the analysis introduces another exponential component to the complexity of the algorithm. (For n parts, m directions and p polygons/part, we get $O(p^2mn^2)$.

One of the goals of the project was to allow the user to load a set of parts (on a standard PC with 1-2G RAM) and with a 'reasonable' wait time (preferably a few minutes) compute a disassembly sequence for the parts. From discussions with the GE Transportation procedure planners we discovered that the designers of assembly, disassembly and maintenance procedures try to limit the procedures to groups of actions that could be accomplished within a single shift. We were told that the typical procedure would involve installing or removing under 50 parts, but this part count did not include fasteners such as bolts, nuts and washers. As such we decided to set our sights on being able to handle assemblies of around 200-300 parts (50 parts with 2 bolts and 2 nuts each). In addition we needed to recognize that aircraft engine parts tend to have complex polygon approximations. The parts tend to have curved surfaces with many irregularities thus requiring hundreds to thousands of polygons per part. Given all of these computation complexity factors we decided to divide the computation into a batch and on-demand component. We

decided a reasonable solution would be to integrate the part pair blocking computation with the PDM and compute this information in a batch mode. The actual searching of this part pair blocking information to find a disassembly sequence for a selected set of parts could then be done on demand.

Our first step to computing the part pair blocking information is to analyze the mating surfaces between each pair of components as follows. We analyze mutual spatial relationship between each pair of polygons taken from each component. A mating condition between two polygons in three-dimensional space is defined, and a list of all polygons in the component that satisfy that condition is generated. Typically, this condition exists if two polygons are parallel to one another and have less than 0.01-inch distance separating them. Once the mating surfaces (polygons) are identified, normal vectors will be applied to each in the mating surfaces' mesh. For each pair of components in the assembly that have non-empty mating interfaces, the interface normals can be used to find the set of all feasible directions for moving one component from another without mutual collision. To determine possible directions of part movement without mutual collision, we compute the Boolean intersection of possible directions corresponding to each polygon in the mating surface and the directional vectors. The possible directions corresponding to one polygon represent a set of points on the discretized sphere that belong to a hemisphere opposite to polygon's normal vector.

Next we create a non-directional blocking graph (NDBG) that captures the constraint information in a concise form. First we need to define a directional blocking graph for a given direction *D* as a list of components pairs. For example, if we have components *A* and *B* in a product assembly such that *A* blocks *B* in a given direction, then the pair *(A,* B) is inserted in that directional blocking graph. In other words a pair of components *(A,* B) belongs to the directional blocking graph for the direction *D* if, and only if, the direction *D* does not belong to a set of feasible removal directions of component *B* relative to mating component A. A non-directional blocking graph is obtained by merging all directional-blocking graphs.

Given the NDBG, we use this information to determine an explosion sequence for the assembly. In essence we use a 'peeling the onion' approach: we find all the parts that are immediately free to move and remove them. Given the current set of parts that have already been removed, we recursively look for parts that are free to move given the current set of parts that have already been removed. At any point in the computation, it may be the case that there are no parts free to move. This can occur if groups of parts mutually block each other in all directions. This will manifest itself in the NDBG as a group(s) of parts, where the transitive closure of the blocking relationship is cyclic. (We call this a strongly connected component in our NDBG.) When such a condition exists, the algorithm removes the smallest strongly connected component as an assembly of parts and then continues on with the recursion.

As mentioned, the sequencing scheme derived above determines a sequence that fully explodes the whole assembly. These types of sequences are useful in the test build, production, and overhaul shops, but are not very useful for on-wing or line replaceable unit (LRU) procedures. In these cases, the objective will not be to tear down the whole assembly, but to remove a small number of parts to be able to get to a specific part. When this is the objective, the sequencing algorithm uses the explosion computation to assign layer values to each part. The parts that come off first are in the 1st layer, the next set of parts are in the 2nd layer, etc. This version of the algorithm looks at the target part and determines the blocked directions that contain parts with the minimal layer assignment. In essence these are the parts that are blocking our target part that are 'closest' to the outside of the assembly. The algorithm recursively starts to analyze the parts in this set to find their removal sequence.

We also recognized that there would always be the need to provide for external conditions that influence the sequence generation that cannot be embodied in the CAD data. For example, it may be understood that an assembly is put together on a table. As such we would want to preclude the sequence generation process from removing parts in a downward direction. Or it may be that there are groups of parts that need to come off as a single unit. We allow the user to modify the NDBG by adding or removing constraints and/or allowing the user to specify parts that are to 'inherit' other parts' constraints.

We had envisioned implementing the sequencing algorithm to perform an exhaustive search on all possible sequences and then apply some heuristics to the solution set to rank the solutions, but after working with the users we determined that they were more in need of an algorithm that reports out the first solution rather than take the time to perform the exhaustive search. We did discover that there was a strong business need for an incremental sequencing algorithm. That is, the users needed to be able to provide a

partial solution for the sequence (perhaps generated earlier from a smaller incomplete set of geometry), and have the sequencing algorithm fill in the 'holes' in the sequence (interleave the removal of new parts as necessary) to make the sequence valid while trying to preserve the initial sequence as much as possible.

Task Details. At the beginning of the program we structured the sequencing research into 9 tasks. The following paragraphs describe these tasks and the effort expended on each.

Create pairwise mating interfaces.

We implemented an algorithm to perform a pair-wise comparison of all the polygons for each pair of parts. When parallel polygons were found (polyA from partA parallel to polyB from partB), the algorithm computes the mutual blocking information as follows: for partA the algorithm determines the hemisphere in the discretized sphere (see 2.3.3.2) that corresponds to partA's normal. The discretized sphere's vectors that are in this hemisphere describe how partA is blocked by partB. The reciprocal is computed for partB.

As mentioned in the earlier discussion, a parallel condition between two polygons is recognized to exist if two polygons are parallel to one another (have normals that are parallel) and have less than a predefined distance separating them on the models. Our implementation of the algorithm used a parameterized value for the distance. After repeated informal experiments with the GE Transportations aircraft engine geometry we decided to set the default value to 0.1. We recognize that the choice for the best distance value will be based on the geometry's scale factor. As such this value should remain a parameterized value. Additional effort could be invested into developing techniques to automatically analyze the overall geometry for a given design to set the distance parameter.

It should be noted that we also added the following extension. If the polygonal normals were sufficiently askew from the discretized sphere vectors we would add the new normal to the discretized sphere, for future analysis involving that part. The measure for adding a new normal was parameterized like the distance measure.

Since this is an $O(n^2m^2)$ algorithm (for n parts and maximum m polygons per part), we compute it in a batch form and store the pairwise blocking information in an ASCII file. The idea is that this computation should be coupled with the Product Data Management (PDM) system so it would be updated automatically whenever the CAD geometry changes.

Determine a set of directions for a possible explosion.

A discretized sphere and normal vectors-to-points on the sphere was used to determine possible part movement direction without mutual collision. The outcome is a set of potential directions for components to be used for assembly explosion.

The discretization we chose to use for the majority of our analysis was in the 6 directional axis vectors (x, -x, y, -y, z, -z). With this the algorithms analyzed 6 directions for each part pair. As with the mating interface distance, we parameterized this discretization. If this discretization is too coarse the next logical step would be to discretize the sphere on the 45 degree angles. This would result in increasing the directional vectors from 6 to 20 vectors. Continuing on to 22 degree angles increases the directional vectors to 120. Other options would be to recognize that GE Transportation models its engines along the z axis. As such parts tend to be removed more on an x-y plane thus the discretization could have a finer granularity around the x-y plane and be more coarse as the vectors move toward a larger z component.

Since, the granularity of the discretized spherical space can have an exponential influence on the overall analysis more research into determining the 'best' discretization approach for a given set of geometry should be considered.

Build a non-directional blocking graph.

The set of potential directions for disassembly generated with the discretized sphere was used to generate a list of part pairs that block one another, relative to direction. For each pair of parts, we analyzed the mating interfaces generated to arrive at a non-directional blocking graph (NDBG).

Once we had implemented the analysis previously described, we discovered that additional analytical techniques were also needed to produce good results. In addition to computing the NDBG from the mating

interfaces, we also added the ability to adjust the NDBG information based on recognizing penetration conditions in the original CAD geometry, and based on additional input from the user.

As we started applying the NDBG computation to the real life geometry from GE Transportation's engine designs we discovered that it is common for parts to be defined such the geometry of two parts penetrate each other. We discovered that these conditions come from approximation errors that result in converting the part geometry to a faceted format, and they could also come from the original geometry itself. The geometry for each part would be accurate, but its placement in the 3D assembled space would position the parts so that they were partially embedded in each other (especially with fasteners – nuts, bolts, washers, etc.). It turns out that the business would use the 3D assembled geometry to study large part collision detection and as such their positions would tend to be accurate. But, the 3D placement of the smaller parts, such as fasteners, has historically been more to produce good images than to perform any 3D analysis and as such, the accurate placement of these parts has not been critical.

We took a 2-pronged approach to solving this problem: an algorithm to try to automatically correct the facet approximation error, and allowing the user to modify the blocking information after it has been computed. The algorithm to 'fix' the facet approximation errors, basically tries to shrink convex and concave surfaces. Thus cylindrical surfaces such as bolts are made narrower and circular holes such as boltholes are made larger. The result of this analysis is stored as add-on information to the NDBG as specifications to ignore part blockings.

The ability to add or delete part blockings was implemented with a user interface that allows the user to visualize the part blocking information. We recognized that for a typical assembly, the part blocking information would be an onerous amount of information to peruse, so we devised a use case scenario that would help the user focus on the part-pair blockings in an intuitive manner. In the visual environment, we allow the user to select an initial part, then color all the blocking parts to highlight that they block the selected part. The user can then select a specific blocking part for further analysis. When the second part has been selected the user is presented with those 2 parts (the blocked part colored blue and the blocking part colored gold) and red and green arrows indicating the part blocking information (Figure 1). A red arrow indicates that the first part is blocked by the second part in that direction. A green arrow shows where there is no blocking. Selecting a red arrow will delete a block (the red arrow turns green). Selecting a green arrow inserts a block (turning the green arrow red).

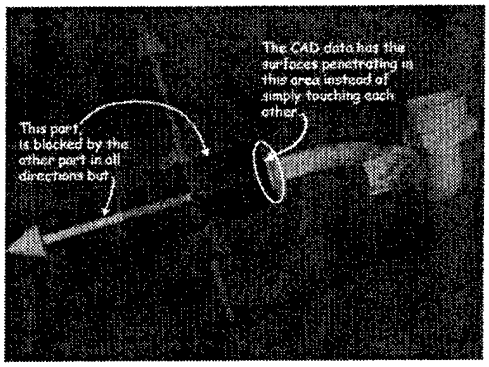


Figure 1 User Constraints Graphical User Interface (GUI)

In addition, the user can do collective blocking modifications based on the state of a sequence. If a sequence is present, then when the user selects the initial part, the display presents the set of arrows that summarize all the blocks from all the parts that have not yet been removed by prior sequence steps. (This display will only contain red arrows.) The user can then delete any of these blocks by selecting the arrows. In this scenario, when an arrow is selected, all the blocks for that part in that direction, with the remaining parts in the sequence are deleted. As with the part penetration fixes, these block edits are stored as add-on information to the NDBG as specifications to add or ignore part blockings.

We also gave the user the ability to specify groups of parts that were to be move as one. These group definitions in essence specified that all the parts in a group were to inherit each others' part blockings. Again this information was implemented as add-on information to the NDBG.

Build the subset of potential sequences and apply assembly heuristics.

As documented above, we use the NDBG information to determine an explosion sequence for the assembly. In essence we use a peeling the onion approach: we find all the parts that are immediately free to move and remove them. We recursively look for parts that are free to move given the current set of parts that have already been removed. At any point in the computation, it may be the case that there are no parts free to move. This can occur if groups of parts mutually block each other in all directions. This will manifest itself in the NDBG as a group(s) of parts, where the transitive closure of the blocking relationship is cyclic. (We call this a strongly connected component in our NDBG.) When such a condition exists, the algorithm removes the smallest strongly connected component as an assembly of parts and then continues on with the recursion.

The sequencing scheme derived above determines a sequence that fully explodes the whole assembly. These types of sequences are useful in the test build, production, and overhaul shops, but are not very useful for on-wing or line replaceable unit (LRU) procedures. In these cases, the objective will not be to tear down the whole assembly, but to remove a small number of parts to be able to get to a specific part. When this is the objective of the sequencing, the algorithm first computes the explosion sequence and uses the explosion computation to assign layer values to each part. The parts that come off first are in the 1st layer, the next set of parts are in the 2nd layer, etc. This version of the algorithm looks at the target part and determines the blocked directions that contain parts with the minimal layer assignment. In essence these are the parts that are blocking our target part that are 'closest' to the outside of the assembly. Moving these parts out of the way will allow the target part to be removed in that direction. The algorithm recursively determines a removal sequence to remove these parts.

We recognized that at many points in the computation the algorithms will have to choose which parts to recurse on first, or which sets of grouped parts to analyze first, etc. We had envisioned implementing the sequencing algorithm to perform an exhaustive search on all possible sequences and then apply some heuristics to the solution set to rank the solutions, but after working with the users we determined that they were more in need of an algorithm that reports out the first solution rather than take the time to perform the exhaustive search. We found that the major use of the sequencing algorithm was to save the user the tedium of defining many part removal paths and more importantly, specifying an animation of these removal paths. In addition, we recognized that even with an exhaustive search of the solution space, we would need to allow the user to modify the generated sequences. As such, we incorporated a simple sequence editor into the evaluation of the algorithm. This editor allows the user to visualize the sequence, add, merge and delete steps and alter the part paths. We found that with this interface the best approach, within the scope of the program, was to allow the user to iterate over generating a sequence, modifying the blocks and/or sequence steps then regenerate a sequence.

We also discovered the strong need for an incremental sequencing algorithm. That is, the users needed to be able to provide a partial solution for the sequence (perhaps generated earlier from a smaller set of the geometry), and have the sequencing algorithm fill in the 'holes' in the sequence (interleave the removal of new parts as necessary) to make the sequence valid while trying to preserve the initial sequence as much as possible. As such we modified the algorithm described above as follows.

When the algorithm finds the next sequence step, it would investigate the supplied sequence to determine if any of the parts being removed by the new step are specified to come off later in the sequence. When such a condition exists, the algorithm would not include such parts in the current step. If it were the case that that was the only part in the new step, this would indicate that that part had to come off at this point to satisfy the restrictions imposed by the geometry. As such, the generator would insert the removal of this part at this point in the sequence, even though the supplied sequence indicated that it should be removed later. In essence, the sequence generator would move the step up toward the front of the sequence.

We could have chosen to let the step ordering in the supplied sequence prevail over the geometrical analysis, but the main call for the incremental sequencing was to be able to regenerate a sequence close to the original sequence, but one that would adjust to changes in the part designs. As such, the validity of the sequence was paramount.

The algorithm described above only generates part removal sequences. The paths show the parts coming off and each step is tagged as a 'part remove' step indicating that for the animation the part(s) should first

appear in its installed position, the animation would be generated showing the part(s) flying off into space, then the part(s) would disappear at the end of the step.

To handle part installation sequences we added the ability to 'reverse' the sequence. But, our reversal is not a simple straightforward reversal. For all sequences, we consider the first step as a special step that defines all the parts that are to be visible in an installed position at the beginning of the sequence. The reverser takes a removal sequence and computes a list of all the parts that are removed in steps 2 through the end. (Assuming step 1 is the beginning step.) It then removes all of these parts from the beginning step (making them invisible), and reverses all the remaining steps and all the paths at each step and tags each step as an install step. The install tagging causes the animation to start with the part invisible at the beginning of the step, then as soon as the first position in the path is rendered, the part becomes visible, it flies into its installed position via the reversed removal path and stays visible at the end of the step.

Investigate symmetric constraints in sequence of events.

We originally envisioned that additional improvements to the sequencing logic would be realized if we analyzed pairs of the CAD geometry looking for axis and planes of symmetry for part pairs. The idea was that this analysis would provide refinements to the blocking information and thus yield better sequences. We did implement a very simple view of this analysis with the mating interface analysis but after investigating the typical geometry encountered in an aircraft engine design, we decided not to pursue it any further. As we investigated this task we realized that the complex shapes of aircraft engine parts and the fact that symmetries are rarely present in the part pairs would make it very difficult to automatically recognize symmetries. We recognized that the effort required to make headway on this task would outweigh the advantages of the analysis within the scope of the program. As such, we decided to redirect our efforts to the additional analysis described in the sequencing algorithm.

Investigate the sense of multidirectional explosion.

Luckily we scheduled this effort to start later in the program because by the time we reached this effort, we realized that like the symmetry investigation, there was not as strong a need for this task as originally anticipated. The reason for this is based on how the technology will fit into the overall engine design, build, and deploy business process. As mentioned above, over the course of the program, we came to realize that initially the maintainability analysts would want to use the immersive end of this technology to assess the larger part removal scenarios and that the sequence generation was wanted primarily to fill in the details of the smaller and much more numerous parts (such as fasteners). We found that it was primarily the large parts that required multidirectional removal paths. The smaller parts most often could be removed from the assembly with a simple linear path.

After an initial investigation into multi-directional path planning we realized that it would take more effort than we had time or funds for, to design an algorithm that could automatically compute a solution (within a full sequencing effort) while the user waited. When we coupled this realization with the reduced need for the algorithm we decided that we needed to modify our approach to this task.

We had a pre-existing batch algorithm (technology from a prior project) for computing multi-linear removal paths. This algorithm took a guide path, containing at least a collision free start and end position, and any number of collision free ordered intermediate positions, and computed, in a batch mode, a collision free path that would move the part from the start position, through the intermediate positions to the end position.

We decided to extend this algorithm and incorporate an interactive version of it into the overall solution. As such, we developed a user interface that allows the user to supply the start, intermediate and ending positions. As the multi-linear path is being computed, the progress of the computation is presented to the user. At any point in time the user can halt the analysis and provide additional guides to the algorithm (by adding or deleting path position nodes).

We implemented this capability independent of the automated sequencing, but envision it as the full solution to incorporating multi-directional explosion paths into the automated sequencing as follows. The sequence generation algorithm could be extended such that if it has exhausted its search for a linear removal path for an individual part, before it starts grouping parts, it could iterate over each removal direction using the extended multi-linear removal path algorithm to search for a removal path in that

overall direction. Since this could be an extremely expensive computation, we recommend that the user interface reflect what is happening with the computation and provide for the user to interrupt the algorithm and give it input. That is, as the multi-linear path algorithm is progressing, it should update the display to show the user the path it has computed so far. An interrupt button would be provided that would allow the user to then supply additional position nodes to speed up the computation or even halt the computation altogether, indicating that the part should not be removed at this time.

Improve and implement/enhance performance of existing explosion distances.

The basic algorithm implemented for this task can be described as follows. First we assume that all parts in the assembly are being removed. (If this is not the case, simply add a last sequence step that contains all the parts that are left at the end of the sequence.) The parts at the last step in the sequence are not moved. Starting at the second to last step of the sequence working forward to the first part(s) to be removed we compute the shortest distance to move the step's part(s) in the specified direction such that the moved part(s) will end up in a collision free position compared to the other parts already removed and excluding those parts appearing before it in the sequence. In essence we compute the ending position for the last part(s) to be moved such that that part is as close to its original assembled position as possible but is now in free space. Then assuming that the removed part(s) is in its removed position, analyze the next to last parts and determine the minimum distance to move it so that it is not in collision. We continue on till we have computed final collision free position for the first step in the sequence.

The logic for determining the shortest distance to move each part(s) is as follows. We compute the length of the line segment resulting from running the part motion vector through the oriented bounding box of the part(s) being removed. We find a collision free position on the removal vector that is an integer multiplier of the line segment length. We then use a binary search to 'bump' the part(s) back and forth on this trajectory. Starting at the collision free position out on the removal vector, we move the part(s) ½ the distance back toward its original position and save this ½ distance value as the current distance value being analyzed. If the part is in collision then we move it ½ of the current distance value back out on the trajectory. If the part is not in collision we move it ½ of the current distance value back in on the trajectory. We then save this ½ value as the new current distance value and repeat the analysis. We terminate the analysis when the change in the current distance value reached some parameterized minimum value.

Develop understandable mating lines for exploded parts.

When this task was originally conceived it was based on the assumption that the technology would be producing static exploded parts images for an IPC. As the program progressed we came to recognize that the technology being developed under this program has the potential for making IPC's obsolete. The main purpose for IPCs is for parts identification and presenting a summary of all the parts in an assembly. The presentation of all the parts in the assembly is in the form of a text table listing the parts, their quantities and their effectivity information. IPCs use effectivity information to present many different configurations of an assembly in one document. IPC's have evolved to the state that we know them today for many reasons, but one of the main contributing factors is the cost of producing, maintaining and distributing individual IPCs for each configuration.

With our technology it would become cost effective to generate a unique IPC for each configuration. We can take this one step further though and realize that if the text part quantity information were included in the procedure instructions and if each frame in the animation documenting the procedure was fully hot spotted with parts identification hyper-links (pausing the mouse over any part in an image causes part identification information such as part number, to be displayed and/or selecting a part number from the text generates a pointer on that part in the image.), then the traditional IPC's do not present the user with any additional information. As such we realized that this technology could well eliminate the need for IPC publications.

Given this, then we decided to revise the objective of this task. Rather than generating understandable mating lines for exploded parts, the problem was reduced to display path trajectory lines in the display as a part was being moved (once the path animation was complete, the trajectory line would disappear). We implemented this capability and quickly found that in some cases it was very useful, but not in all. In the sequences where there were numerous parts moving at once in a single step, if many parts were involved,

the trajectory lines could become numerous and be very distracting. We addressed this issue by parameterizing the path display feature.

As we developed sample sequences and prepared to use them on the assembly floor, we experimented with turning the path lines on and off. We have come to the conclusion that a hybrid approach would be the best solution. In the hybrid approach, if the sequence were an install sequence, the display would initially show the full path. As the part moved the path segments that have already been traversed would be deleted from the display. If the sequence were a remove sequence, initially no path segments would be displayed, but they would come into view as the part moved over each segment.

At this point, a simple switch in the program turns on and off the full path display for a part and its corresponding path. We recommend extending the algorithm as described in the preceding paragraph.

Develop automated explosion animation.

Our effort on this task can be summarized as follows. First we added the ability for the user to add an ordered list of camera angles to the beginning of each step. The animation will first fly through the set of camera positions at the beginning of the part removal animation for that step. This will allow the user to define close-up shots on parts and to position the camera for the best viewing of the part removal animation.

As mentioned in the sequencing algorithm description each step carries a tag indicating if it is an install or remove step. The install steps are animated such that the part is originally invisible, it becomes visible as it flies through its install path and stays visible after that. The remove steps are animated with the part originally visible; it stays visible as it flies out on its removal path, and then becomes invisible at the end of the step.

To generate the animation the algorithm generates a series of jpg images and then passes these images to a public domain filter to generate an avi animation. For each step we generate a set of jpg images for each camera angle and then if there is a removal path, we generate a set of jpg images capturing the intermediate segments on the removal path. The user has the ability to add and delete the intermediate segments, thus allowing them to control the 'speed' of the path. If there is no removal path, a series of jpg image are created showing the part in place, then showing the part removed (if it is a install step, then the first set of images are without the part and the second set are with the part installed). This in essence adds a pause to the animation at this step, producing better animations when the sequence has a mix of steps that have removal paths and steps that do not.

We purposely wanted to keep the animation definition as simple as possible. We do not have all the timing mechanisms that one would see in a commercial animation package, but so far we have been able to use the concepts from above to generated very acceptable animations.

3.3 Task Generation

Overview. This task was responsible for developing techniques to produce a comprehensive bundle of sequence animations and all the 'other' text information that goes into a technical manual (such as descriptions of the actions, warnings, notes, cautions, etc.). We call the 'other' text information *domain information.* It is important to note, that we are not trying to automatically generate the published technical manuals, instead we are trying to develop techniques to automate the bundling and organizing the technical content for the manuals. In some cases, such as the ordering of the sequence steps and the animations, the content is generated, in other cases, such as with the domain information, the content is retrieved from some central store. The intent is that the tech writers will take these bundles of information, bring them into their publishing system for the final formatting and distribution of the publication.

At the onset of the program we anticipated that the majority of the effort would be put into generating the description of the actions. The overall concept was to generate these descriptions and couple them with the warnings, notes, cautions, torques, tools, consumables, etc. that were appropriate for the action and also bundle in the imagery generated from the 3D CAD. The ultimate goal was to generate a comprehensive set of content for inclusion in a technical manual.

As we became more familiar with the state of the Technical Publications technology and the possibilities of this program we realized that there was not a strong need to generate a detailed description of each part

motion, but there was a strong need to develop a characterization of the domain information that would support retrieving this information at the granularity necessary to support the sequence steps.

We initially had assumed that there would be databases of the domain information for us to draw from. We had assumed that this organization of the domain information has occurred as a result of moving to the IETM (Interactive Electronic Technical Manual) technology. We were mistaken. As we became more familiar with the current state of the technical publications technology we discovered that typically IETM pieces were far too high level of granularity for our purposes. IETMS are developed to be reusable 'chunks' of a technical manual. The intent is to support inserting these chunks into different contexts, but typically, the smallest form of an IETM covers a couple of steps in a procedure (if not the whole procedure) and has the domain information already bundled in with it. From an analysis of our needs, we determined that we needed to be able to find domain information that ranged from being applicable to an individual action in a procedure (such as remove a part) to being applicable to any reference to the whole engine. We needed to be able to find domain information with queries such as: "Find all warnings that are general to partA", "Find the torque to be used on partA when it is being installed in position x", etc. The IETMs organized their information at a much higher level of granularity and thus we were not able to use the IETM strategy as our notional domain information database.

With regard to the detailed description of the part motions, we realized the adage "a picture's worth a thousand words" is really true. We studied existing technical manuals and realized that the detailed descriptions are necessary only because there is no other mechanism to convey the action in the technical manuals. An animation of the sequence itself is a more comprehensive, less ambiguous and faster to comprehend description than a written description. Animations typically are not included in a manual because they are too costly to generate and maintain.

As we studied existing manuals we realized that if we coupled a step-by-step animation of the sequence, a summary description of each step, and a structured presentation of the additional domain information, a better, easier to comprehend technical manual would result. We developed a prototype presentation of this information and showed it to technical writers, assembly planners and mechanics actually performing the procedures from both GEAE and Lockheed Martin. The feedback was unanimous that the animations and more succinct text material were easier to comprehend.

From this verification of the approach we develop a detailed characterization of the information that needed to be bundled and thus established the access requirements (indexes and retrieval schemes) for collecting the domain information for the procedure. We concluded that the animation of the sequence needed to be able to provide one continuous 'play-back' of the sequence as well as a step-by-step breakdown of the sequence. We also recognized that if we could automatically hotspot the generated images with parts identification information then we would have significantly improved the usefulness of the images and taken a very large step toward outdating the current IPCs and 2D drawings. In addition, we realized that the different types of information required different retrieval schemes. For example, for a sequence step instructing the user to remove partA, we want to show any warnings, notes or cautions that are associated with

- the overall engine line,
- the overall assembly,
- any instance of this procedure,
- generally to the part,
- specific to the part in this context (engine, assembly, procedure),
- general to the action on the part, and
- specific to the action on the part in this context (engine, assembly, procedure).

This implied that when generating a sequence (*sequenceA*) for a given part removal or installation we needed retrieval mechanisms that would issue multiple queries. As in the case of warnings on a part removal (partB), we would want to query for warnings that:

- apply to any procedure involving the engine involved in *sequenceA*,
- apply to any procedure involving the assembly involved in *sequenceA*,
- apply to any instance of *sequenceA*,
- apply to any uses of *partB*,

- apply to any uses of *partB* in *sequenceA*.
- apply to any removal of *partB*, and finally,
- apply to any removal of *partB* in *sequenceA*.

A different retrieval scheme is required for torque information. For torques, the retrieval needed to look for a torque that had been defined as specific to the instance of the part, or if such a specific torque was not defined, then to look for a torque that applied to any occurrence of that part in the specific assembly, or finally to find a torque that applied to any use of the part.

In summary, for certain domain information, we realized that we would automatically issue broader and broader queries and form the union of these queries, and for others we would only broaden the query till the desired information was found. We called the issuing of broader and broader queries, query expansion. For each type of domain information we characterized and implemented the appropriate query expansion mechanism. A detailed description of the data types, the required indexes and their query expansion mechanisms are documented in Appendix B.

As mentioned earlier, the overall goal of this area of research was to generate and bundle the critical information for the technical manual. As such we designed an open intermediate data structure for capturing and organizing the bundle of information. We called this intermediate data structure the Task Generation Logical Form (TGLF). The TGLF is purposely presentation neutral. It simply embodies a step-oriented organization of the sequence data. It has been implemented as an XML Schema Definition (XSD). For demonstration purposes, we also developed two transformation files (xslt) to take a TGLF file and transform it to a web page representing a notional technical manual page. One transformation file generated pages describing the procedure and another transformation generated pages similar to an Illustrated Parts Catalog, IPC (Figure 2). We also implemented the ability to generate an avi file that presents an overall animation of the entire procedure, a sub-animation of each step, and the ability to stop on any frame in the animations and have every part visible in the frame contain a behind the scenes 'hot-spot' definition so as to support interaction with the user based on the user 'mouse-ing' over the parts in the image.

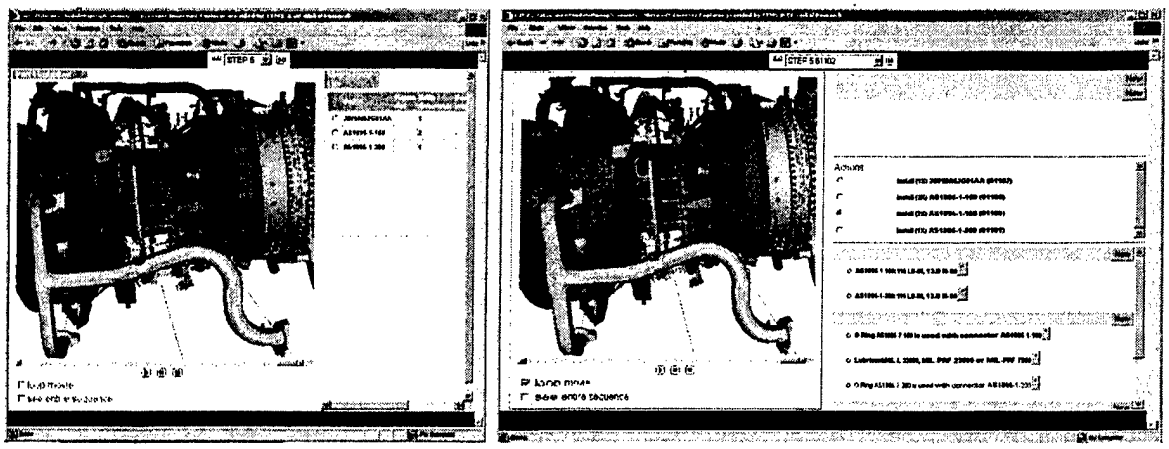


Figure 2: Generated Content, IPC and Procedure Style Presentations

The IPC generation is interesting in that it demonstrates that this new technology may change the very approach to organizing IPCs. Historically the business would publish one IPC for all the documentation on the engine. The IPC for the engine would provide an assembly-to-subassembly drill down through the parts in the engine. The IPC's are primarily used as a parts identification tool, and a configuration management tool. As changes are made in the engine design, the IPC would be updated to reflect these changes. Typically these updates were in the form of notes indicating a part number change and the change's effectivity information. For the most part, the artwork would not be changed because it was too cumbersome to manage all the possible combinations of parts. As a result, understanding how a certain configuration should look could sometimes be confusing. In addition, the IPC presented a single drill down through the engine parts that were independent of the other documentation (there was not an overhaul IPC, an inspection IPC, etc...) It is not uncommon to have the documentation for a specific procedure span

multiple assemblies, and thus reference multiple IPC images. (In such cases, the user would have to look at each image and build their own mental union of the images.)

We now realize that with the automation being developed by this program, IPC pages can be generated specifically for a given configuration and a given procedure. It is the automation that will make these procedure specific IPCs cost effective. It may even be the case that it will be cheaper to use the technology to generate procedure specific IPC's than to use the technology to generate traditional IPCs.

Task Details. The following pages describe the 5 major tasks we undertook to develop a solution to the information bundling.

Design or adapt language for primitive actions.

The goal of this task was to develop a data representation, the TGLF, which would represent the task corresponding to the service sequence. Sequences, primitives, and associated domain information were embodied within the data representation.

The TGLF, which can be thought of as a slot filler or frame type data structure that embodies task order concepts such as actions, orderings of the actions and targets of the actions. The TG Logical Form is populated from the service sequence, associated images, lexicons, and available domain information.

The TGLF has been implemented with an xml schema definition (xsd). It is documented in Appendix A.

In addition to the TGLF we characterized the domain information that is present in a technical manual. We used sample technical manuals from GE's LM2500 engine (a marine engine), the CF34 engine (a regional jet engine) and the GE90 engine (a large commercial engine). With the FAA regulations, we found that the technical manuals were well structured, making it relatively straightforward to characterize the types of information in the manuals and allowing us to feel confident that we were covering all the data types represented in the manuals. The characterization of the domain information appears in Appendix B.

There was one type of information that the research did not cover, namely procedure steps that did not involve moving a part that had been modeled in the 3D CAD environment. Such steps typically involved positioning flexible parts (wiring), smaller consumables such as oRings, or inspection steps. The TGLF can handle these steps, its just that the sequence generator only generated steps based on the 3D CAD data. A full production version of this software must allow the user to insert non-CAD oriented steps.

Translate or convert exploded view into language of primitives and insert human actions into the translated sequence.

The goal of this task was to populate the TGLF with information from the service sequence, primitive operations, and available domain information.

Populating the TG Logical Form was implemented as a process that uses an XML parser to recursively examine each step within the service sequence. As each step is processed, primitive actions and domain information are inserted into the logical form structure.

A Noun-Verb pair and the task type are the minimal required information for populating a TG Logical Form. A partmap file that maps each part number from the service sequence to a part name is also required in order to accurately retrieve domain information.

As previously described, we extended the domain information characterization with a query expansion technique. For each information type, we characterized how the retrieval mechanism needed to operate in terms of being able to expand its search for relevant domain information. The query expansion details are also documented in Appendix B. In essence the expansion operates on a prioritized combinatoric key expansion approach. With this, two types of keys are supplied to the query engine, absolute keys and a prioritized set of key. Queries are generated such that they look for data index by the absolute keys and all combinations of the prioritized set of keys. The order of the queries is governed by the prioritization of the keys. For instance if the procedure code, assembly, and engine line were the prioritized keys (prioritized in that order), and part number was an absolute key, the search engine would generate an set of queries looking for information indexed as follows (in the order shown):

- part, procedure code, assembly, and engine line,
- part, procedure code, assembly

- part, procedure code, engine line,
- part, assembly, engine line
- part, assembly,
- part engine line,
- part

In addition, we also needed to characterize how and when to stop the expansion. This boiled down specifying if the query was trying to find a single piece of domain information or the union of all the relevant domain information. If it were trying to find a single piece, then it would stop once a query returned a value. Otherwise it would perform all the queries and return the union of all the results.

We also found that a single part may actually be represented with multiple part numbers. For example a part may be given a specific part number designation in engineering at design time, but the assembly and service arena may use the suppliers part numbers. In GE Transportation, even in the design area, each part could carry up to 3 different forms of identification.

At GE Transportation, each part always has a given part number, but within an engine configuration, the team would also assign a part position number (ppn) to every part in the design. PPN numbers were used to reduce the impact a design change had on the documentation. The documentation for the design would contain a mapping from ppn to specific a part number. The artwork and text for the design would reference the parts by ppn. As the design changed and one part was replaced with another, if the image of the part was still similar to the original part, the ppn to part number mapping would be updated, but the artwork would not be updated. As such, the ppn and part numbers both identify the parts. In addition, it is often the case that a part is used multiple times in the product. When this occurs there is the need to be able to identify each use of the part. (This is especially true with life-limited parts.) And so, a part instance number is tacked on to the part number producing in essence a third option for identifying a part.

These alternate part-numbering schemes also need to be incorporated into our query expansion mechanism. The most specific query would look for information tagged to a specific instance of a part. The next levels of expansion would iterate through the ppn, the design part number and the supplier part numbers (in that order). As such we built a part mapping capability and incorporated the alternate part number prioritization scheme into the query expansion. With this, the expansion operates as described above, but when a part number appeared in any query, that query would be expanded to substitute the alternate part numbers, in the order just mentioned.

Reduce subsequences of primitive actions in task-order steps - basis for sentence generation.

The goal of this task was to integrate and refine the TG Logical Form such that the primitive steps are coherent in the context of the larger structure of a task order and the task structure.

The service sequence, from which the TG Logical Form is generated, represents a sequence of steps for a particular task in a certain context. For example, we had expected to find differences in the language, procedures and steps that are followed when performing a task at the on-wing level versus the depot level. And we anticipated embodying the differences within an associated higher order Task Structure, so that the differences could ripple down into the text that is generated from the TG Logical Form.

For this task we used a number of technical documentation domain experts and analyzed the differences between the different forms of technical documentation: test assembly, production assembly, overhaul, and on-wing. We found that the specific context impacted the procedure steps more than it impacted the text describing the procedure. An on-wing procedure would be defined to remove as few parts as was practical, overhaul procedures tended to be full tear down and build ups and the test and production assembly procedures were most often designed as staged build-ups of sub-assemblies to assemblies. Once the appropriate ordering of the parts was established the text that was used to describe the action on the part was common across the contexts, typically involving 5 main verbs: install, remove, inspect, clean, and measure.

As a result of the analysis we realized that the operations using the technology would realize the most benefit by taking the context into consideration at sequence design time. For aircraft engine design, higher order task structures are well defined. In the commercial world the ATA structure defines this higher order

task structure and the military engines have their own Logistic Control Numbers (LCN) that contains a decomposition of engine areas and procedures.

As we investigated the business procedures we found that as soon as the CAD data was generated, the designers would start compiling groups of parts for assembly oriented analysis – answering questions such as, can I access this part, can I see this part, etc? Unfortunately the assemblies used in these early analyses tended to be ad-hoc set of parts the designer decided to load. We realized that if the part sets in the early analysis were aligned with the part sets needed in the eventual technical documentation, then the imagery, sequences, etc. from the early design analysis would be more useful in the downstream business process. In addition, if the technical publications context structures were adopted further up in design, then an automated mechanism could pre-compile the parts sets and the upstream organizations could benefit from having the part sets already defined for them (they would not have to compile the parts themselves).

The prototype software demonstrating this program's technology supports the manual compilation of these part sets. We highly recommend that effort be invested in tools and developing standard business practices for organizing the design parts so as to align the design analysis with the downstream uses of the data (technical publications, service, etc.) as early in the design process as possible.

Generate human natural-language statements to be put into haptics verification system.

The goal of this task is to compose multimedia information bundles (text, graphics, and hyperlinks), which can be used for presentation for a wide variety of applications. This task will provide the overall structure and mechanisms for accessing the TG information as well as utilizing context and providing for presentation. This will result in transforming the internal representation (the TG Logical Form with associated Task Structure) into user output such as in the haptics virtual validation system or as data input into an authoring environment.

The TG Logical Form and Task Structure, which was defined in XML, can be transformed to another format using XML transformations through the use of XML style sheets. By applying style sheets to the logical form, readable and understandable text-based instructions can be generated for a variety of applications. We demonstrated this capability by creating 3 forms of style sheets. Two of these forms present the information from the TGLF in an organization similar to a procedural manual and the third in a format similar to an IPC. Figure 2contains images of these presentation formats. The two that present the information as procedural documentation differ in that one also incorporates add and delete buttons that couple the presentation of the material with a notional domain information editing system. This demonstrates a process flow where the sequence generation generates the content, then presents the content in a manner that allows the user to modify the underlying domain information knowledge base so that the next pass through generating the content will produce better results. The second mechanism for presenting the procedural style content does not have the add/delete buttons. This represents a read-only view of the material. The third presentation style, demonstrates that the TGLF can support the information necessary to present an IPC.

The style sheets we used to produce the various views of the TGLF demonstrate how the XSLT technology can be used to simply format the information, or more powerfully, to re-order and summarize the information. In the procedural view, if there are multiple instances of the same part number being removed on the same step, we combine these instances into a single statement.

The information bundles will be integrated for presentation and interaction with the haptics verification system.

The goal of this task was to develop an interface within the haptics verification system so that the information bundle for the task may be effectively presented and navigated for tech order validation.

At first we envisioned using speech output, but by the time we turned our attention to this, we recognized that the industry was well on its way to solving the text-to-speech problem, so rather than focus on speech output; we focused on integrating the navigation of the natural language instructions with the user interactions in the virtual validation environment. Therefore, within the context of a higher order task structure and processes, we explored: 1. the presentation aspects of the text and graphics in the virtual validation environment and 2. coupling the navigation of the information with the user interactions already in the virtual validation environment.

In 2001 we developed a usability analysis of how the information should be displayed within the haptics environment. The intent was to use these guidelines as an initial starting point for presentation within the haptics environment so that the integrated technologies could be used to validate the procedures.

In 2002 we expanded this concept to include a description of how the sequence generation would fit in and how the 3 integrated technologies would be incorporated into the business processes. As we became more familiar with the possibilities of the technology and its use in the business processes we came to realize that the concept of using the technology in technical publications to perform virtual validation would require a larger change to business operations than could be realized within the term of the project. Instead, we realized it would be better to target initially inserting the immersive technology into the procedure development process, well before technical publication's participation in the design, build, and support process. We came to this conclusion based on a better understanding of the state the technology could achieve by the end of the program, the business process, and the business funding structures. The primary driver was the business process. It turned out that once they saw the technologies, the maintainability analysts and procedure planners realized that the immersive and sequence generation technologies could have a very large, positive impact on their operations. The technical publications groups also saw the power of the technology, but we came to understand that the true power of the technology would best be realized by using the technology as pervasively in the design, build, and support process as possible. We realized that the technology actually helped a much larger user community than just technical publications, and targeting users early in the process would make it easier to adopt the technology in downstream processes and maximize the impact of the technology.

As such, we decided that the primary goal of this task would be to cover the needs of the 'front line' users of the immersive technology (the procedure planners and maintainability analysts). They would be the ones to use the immersive and sequence generation technologies to create validated sequences. At this point in the business process the main concern of the users would be to validate that the sequence of procedure steps made sense and each part could be removed as described by the removal paths. Once the procedure steps were validated, the final validation effort would be to verify that 1. the appropriate imagery was generated, 2. the appropriate warnings, notes, cautions, etc. were bundled with the imagery and 3. the information was bundled in a manner such that the presentation mechanism could allow navigation through the material as per the users' needs.

So in the end this task boiled down to providing presentation requirements to the virtual validation and sequence generation research areas and defining interaction requirements for the output of this Task Generation technology. The presentation requirements for the SG and VV technologies can be summarized as follows:

- the VV technology needed to support the definition of initial part removal/installation paths and/or sequences.
- The SG technology needed to be integrated with the VV technology in that the SG analysis needed to allow paths and initial sequences generated through the use of the VV technology to be supplied as preconditions to the generated sequence.
- The Task Generation technology needed to allow the user to interact with the generated output (as described below) and have some mechanism for providing feedback when the domain information database needed to be updated and/or images in the output needed to be modified.

The VV and SG requirements are further discussed in VV and SG sections. The TG feedback requirements are covered in the following paragraphs.

The interaction with the generated output can be characterized as interactions with the text information and interactions with the images. We used the strength of the XSLT technology to demonstrate how the TGLF supports bundling domain information interaction with the presentation. Our procedural style sheet generates presentations that are capable of reacting to the user in the following manner. When there are warnings, cautions, notes, etc. that are specific to one of the parts in the step, and when the user selects that part in the display, the presentation automatically adjusts to highlight and make the part specific information more visible (putting it at the top of the list.) In addition, when the user selects a part from an image, the IPC style presentation of this sequence is presented to the user.

In terms of the image interaction, we realized that user needed to be able to 'point' to a part and be given information about that part; in particular part identification information (ppn, part number, etc.). As such

we developed algorithms to take the CAD based image and define region mapping from the bits in the image to the parts being displayed at each bit. With this, we build an image map to hot-spot each part in the image. For each image, when a user places the cursor over a part in the display, the part's ppn, part number and instance is displayed. The concept of hot-spotting an image to display part number information is not new. The technology to automatically hot-spot the image is however. This automation is very powerful in that it will make it cost effective to fully hot-spot every image, thus making it feasible to link what-ever part oriented information is available to the image.

Integrate automated sequence and task generation.

The goal of this task was the development of a prototype system, which can generate instructions based on the CAD geometry of an assembly. As mentioned earlier, the SMG architecture consists of separate, but interoperable, modules for SG, TG, and VV. The modules interact with each other through a shared data repository and through the use of an XML messaging interface.

The SG technology includes the generation of a partially populated TGLF and avi and jpg imagery to animate the procedure. The SG technology builds a TGLF that contains the following information:

- Some high level task context information:
  - A procedure identifier
  - The assembly the sequence is for
  - The engine model and build the sequence is for
  - The overall type of procedure, assemble, disassemble, etc.
- The series of steps that make up the sequence. For each step the TGLF contains:
  - A list of the parts active in the step
  - A tag indicating the overall action: remove/install
  - (Optionally) a removal or install path for each part

The TG process takes the initial TGLF from SG and populates it with the domain information from the shared data repository. For each step in the sequence, the TG exercises the queries described above and places the information back into the TGLF. At the end of the process, the TGLF is passes through a variety of XSLT transformations to generate a self contained directory of html pages that present the TGLF content. The presentation is notional. The ultimate goal for the information is to use an XSLT transformation to generate content that is compatible with the technical manual authoring system.

The avi file (generated in the SG module) depicts the full sequence from the first to last step. The individual jpg's are coupled with a playback mechanism in the presentation that plays one step at a time allowing the user to stop on individual frames of the animation (with each frame fully hyperlinked as mentioned previously).

3.4 Virtual Validation

Overview. The goal of the VV component (Figure 3) was to combine a number of virtual-reality technologies in an effort to provide a realistic environment for determining the validity of the steps within the sequence. These technologies include a stereo Helmet-Mounted Display (HMD) to immerse the user in the environment, a data-glove for modeling the hand in the removal process, a tracking system for mapping the real-world positions of the HMD and data glove to positions in the virtual environment, and a 6DOF SensAble ™ PHANToM™ for enabling realistic human interaction with the virtual environment [5].

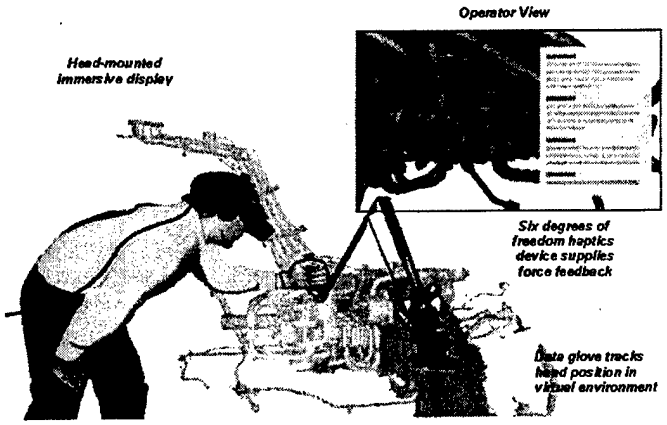


Figure 3: VV system

GE Global Research had been working for some time on the use of haptics for maintainability analysis. Initial investigation for this task began in 1998, when we developed a collision technique based on using points sampled from the surface of a moving part, and a volumetric grid containing proximity/penetration information for the non-moving parts, similar to the technique developed around the same time by Boeing [4]. In the fall of 1999, we delivered a prototype Haptic Path Planner tool to the Joint Strike Fighter (JSF) military engine program at GE Aircraft Engines. This work provided the foundation for the haptic component of the Virtual Validation system in SMG. For Virtual Validation, we needed to enhance our approach to allow for parts to be selected in a random (user-determined) order as the moving part for removal. This allows multiple parts to be haptically removed in rapid succession. In the paragraphs below, we describe in some detail how we represent data in our haptic environment in a way that allows for multiple part removal, and how we perform fast collision detection, independent of the polygonal complexity of the models used in the environment.

Data Representation

There are three main components to our haptic representation. We describe each of these components in turn.

Surface Points. For each part that participates in the removal process, we generate a set of points that lie on the surface of the part which are evenly spaced and of sufficient density to adequately represent the part at a desired resolution. The surface points for a given part are used only when that part is the "moving part" in the environment.

Penetration Map. For each part relevant to the maintenance task, whether it is one to be removed or not, we generate a fine-resolution volumetric grid called the "penetration map", which contains proximity (penetration) information for points that lie outside (inside) the surface of the part, along with surface normals corresponding to the surface polygon nearest to the given point in the volumetric grid. In order to conserve space, this grid is arranged in a two level hierarchy in which a 4x4x4 block of grid points are grouped together to form a "brick". If all the grid points within a brick are greater than some maximum distance outside the part, or greater than some maximum distance inside the part, then the brick consists of a single value identifying that state. Otherwise, the brick is broken down into the 64 grid points with detailed penetration and surface normal information.

Part Map. The collection of (possibly overlapping) penetration maps of all the relevant parts defines a volume of space consisting of the union of all the volumes of the individual penetration maps. This combined volume is subdivided at a coarser resolution than the individual penetration maps to produce a grid called the "part map". This part map contains, at each grid location, a list of which parts are relevant within the space represented by the given part map grid location (Figure 4).

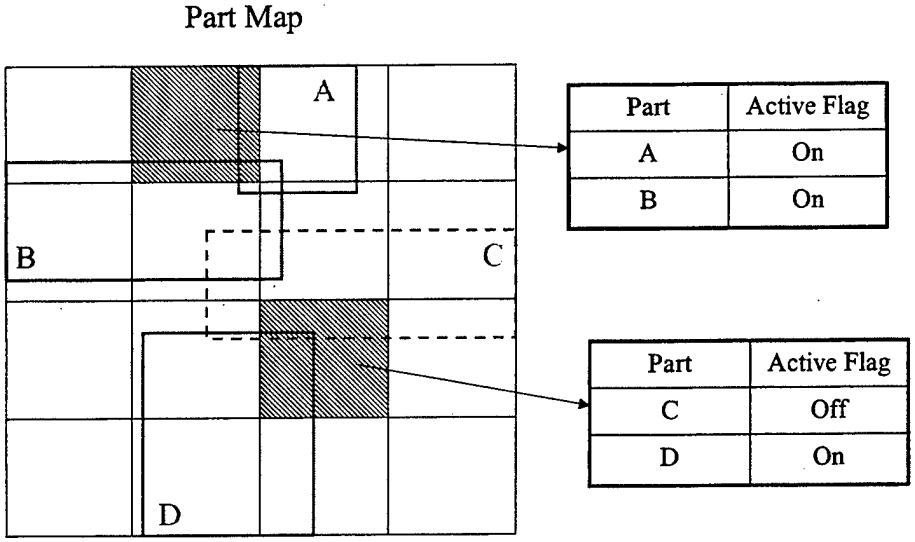


Figure 4: Part Map example

Setup

We select the moving (those that have generated "surface points"), "collision active" (those that have a computed "penetration map"), and visual only parts (all those that do not have a "penetration map") via our user interface. In particular, the user "picks" (typically by pointing and clicking with the mouse) all parts that will be candidates for interactive removal and then uses bounding boxes to select all "collision active" parts (if a part falls at least partially within any of 1 or more bounding boxes, it will be "collision active"; also the candidate moving parts are also automatically "collision active").

Selecting the Part to Remove

The tip of the haptic stylus is portrayed in the scene as a small sword. As the user moves the tip of the haptic stylus the sword within the scene moves. A single click on the stylus selects the part directly beneath the pointer. A double click 'attaches' the sword to the selected parts initiating moving the part via the haptic device motion and performing the collision detection analysis with all subsequent motion of the haptic stylus. Selection via the glove or voice input has also been discussed as a possible confirmation method, but has not been implemented.

When a new part is selected to be the moving part, it is removed from the list of fixed parts; this switching of parts occurs "instantaneously" (no apparent delay to the user).

Fast Collision Detection

At each iteration of the haptic collision detection process, the sampled points of the moving part are transformed according to the moving part's position and orientation in the virtual environment. The resulting location is indexed into the part map to determine which parts are possibly in contact with the given point. Since any one of these parts may have been previously removed in an earlier step of the removal sequence, each part has associated with it an "active" status flag, indicating whether or not it participates in collisions with the moving part (Figure 4).

For each active part in the part list at the current sampled point's location, we conduct a penetration test with that part's penetration map. Based on the sampled point's location, we look up the eight surrounding grid points in the penetration map and tri-linearly interpolate both the proximity/penetration distance and the associated surface normals. We keep track of whichever part the sampled point penetrates the most, and use that part's interpolated penetration distance and surface normal to calculate a collision response.

In order to take advantage of a sufficiently dense set of sampled points without wasting precious CPU cycles on points that have no chance of being in collision, we take advantage of the temporal coherence of the moving part through a technique called "skip counts". Each time a collision test is performed on a sampled point, we determine, based on the distance of that point from the nearest surface and the maximum speed with which we allow a part to move, the minimum number of iterations that must pass before that point can possibly collide with any surface. This number becomes the "skip count" for the given point. At each iteration of the haptic loop, we decrement the skip count of each point for which the skip-count is non-zero, and perform the collision test only for those points whose skip-count is zero.

Task Details. At the beginning of the program, we had already decided to divide the VV effort into a series of 8 tasks. The following pages describe each of these tasks, what our initial vision was for the task, and the results that were achieved as the program progressed and we learned more about the technology and the business drivers for the technology.

Improve state-of-the-art for interaction realism by enhancing haptics feedback

This task encompassed developing the following capabilities:

- Integrating the volume/point sampling technique documented above into the prototype environment.
- Integrating the collision detection analysis with the haptic device to
    - Recognize and react to moving the haptic stylus,
    - Generating and sending feedback forces back to the haptic device and
    - Generally become familiar with effects of the various device parameters and choosing the best set of parameters for this application.

  This work required developing an understanding of the SensAble device library routines and calling the correct routines to retrieve the device positioning and to position the device. Information regarding these routines can be obtained from [5].
- Implementing part dicing, view-frustum culling and level-of-detail rendering to speed up the rendering so as to make the apparent part motion match the speed of the haptic stylus motion. These capabilities are all provided as standard features via the VTK [1] software, we simply had to organize our higher level routines to be able to handle these data structures.

Integrate virtual reality technologies with haptics environment.

Immersive head mounted displays, data glove, and body-tracking points were integrated with the haptics environment. Correction algorithms for specific issues regarding implementation of these technologies for the purpose of maintenance simulation were addressed.

An Ascension Flock of Birds [6] tracking device was mounted on the top of a Head Mounted Display device (HMD). Software was developed to set the camera viewpoint based on the input from this head mounted tracking device. As such the scene being displayed through the HMD (and on the screen) would follow the user's head motions providing the appropriate viewpoints based on the positioning of the head.

The part dicing, view-frustum culling and level of detail rendering mentioned in the previous task was also necessary to achieve rendering speeds that could keep up with real-time head motion.

We also integrated Immersion's CyberGlove [7] with the HMD as follows. We mounted another Ascension tracking device on the back of the glove and used the hand location information from the tracker to position the glove's geometry in the virtual environment. (The technology from the manufacturer already supported moving the CAD fingers to match the wearer's finger motions.) We implemented logic to recognize the glove as gripping an object if the pinky through pointer fingers were bent (off the plane of the back of the hand) at greater than a 60-degree angle.

It was our intention to integrate all 3 virtual reality devices, the HMD, glove and haptic device, but we were not able to achieve that goal. There were no technical barriers to doing this. We did demonstrate coupling the glove and the HMD and coupling the HMD and the haptic device. It's just that as the program matured we realized that the most immediate pay-off for the technology would be with the sequence and task generation capabilities. So, toward the latter half of the project we shifted our focus away from raw technology development and more toward functional demonstrations. We realized that demonstrating an

integrated glove, HMD, and haptic device, though novel, would not have as large a business impact as demonstrating an effective SG/TG capability.

The main work left to demonstrate an integrated HMD, glove and haptic device is to develop an approach to coupling the haptic stylus and the cyber gloved hand. There must be some mechanism for attaching the stylus to the hand, but it must be such that the hand is allowed to virtually 'grasp' the virtual geometry.

Explore and develop methods for multipart removal.

To improve usability of the technology, it is desirable to remove multiple parts from the virtual environment in series without dependence on a preplanned sequence of events. Improved data structures and faster preprocessing was required to achieve this goal.

The data structures and analysis described in the VV overview explains how we implemented the multi-part removal. A drawback to our implementation is that is doesn't support recognizing collisions with moved parts. When a part is moved, while it is moving the technology will recognize when that part collides with other parts. After the part is moved to a new location, future collision analysis with new moving parts, do not consider the parts that have already been moved. The part's active flag is turned off.

To extend the technology to support this capability would require updating the part map after or as the part is being moved. We do not think there is a large technical risk to implement this.

Develop algorithm for correcting positioning of the haptic device.

We had intended to improve the realism and usability, developing techniques for the correct positioning of the haptic device, e.g., moving the haptic device to the user's hand. We did not implement this capability, but our high level design as described below could be used to guide a future implementation.

The haptic manufacturer indicated that there is a routine that positioned the device to the specified coordinates. Thus, the only remaining effort is to develop the interface or logic to determine the desired position and integrate it with this routine. We envision that when this capability is wrapped into the technology, the target position for the device will be determined by the position of the haptic glove. A trigger event that causes the device to be automatically positioned will also have to be developed. Since the intent is to ultimately have the user's hand(s) ready to receive the haptic device(s) we envision this event to be some sort of voice-activated signal. We would expect that off-the-shelf voice recognition tool would be wrapped in with the immersive environment and that many of the mouse-click events in the prototype application could be replaced by asking the user to speak key words.

Incorporate a feedback mechanism from validation system to exploded views and language generators.

This capability was achieved when we decided to support the use of the immersive technology prior to generating the sequence. That is, we developed an interface allowing the user to define removal paths and complete disassembly sequences with the haptic device. With this we developed the ability to define and edit the paths and sequences with the device.

There were 3 more tasks that we had envisioned working on with this effort;

- Emulate kinematic effects in a virtual environment.
- Develop and implement hybrid collision-detection techniques.
- Implement tracking of multiple human events in virtual environment,

but they also fell prey to the higher priority SG/TG needs. With each task, we had a number of ideas on what we needed to develop and how to approach the solution. You will see that we discuss the capabilities targeted with these tasks in the *Next Steps* section below.

3.5 The Demonstration

Overview. At the beginning of the program we had an idea of how to use the technology, but it was based on general assumptions about the business process. By the time we were half way through the program we focused more on the details of how the technology could be used and how it would integrate with and affect the existing business processes. As we mentioned in the introduction, we came to realize that the technology would be immediately most useful to the following groups:

- Maintainability analysis,
- Test engine procedure planning,

- Production assembly procedure planning,
- Service procedure planning, and
- Technical publications.

As the technology becomes more mature additional users will be found in Training, Marketing, Service Shops and even back in the initial Engineering Design areas.

We realized that in order for the demonstration to achieve the most impact we needed to present the technology not from a technology point of view, but from a user's point of view. In other words, we needed to be able to communicate how the audience would use the technology in their day-to-day work. As such we decided to wrap and present the technology as a prototype maintenance/procedure planning tool that could generate instructional content from the planning results.

With this approach we demonstrated how a maintainability analyst would use the tool to perform an initial maintainability assessment of a design. We showed how the technology facilitates performing the analysis, capturing the results so they can be later used as the initial definition of the maintenance and training procedures. We also demonstrated how to apply the technology to the assembly planning for the test build engines and use the results of this use as the initial content for the technical publications authoring process.

The following paragraphs give a broad brush description of the major functional areas in the demonstration prototype. More detailed information appears in Appendix C, the user manual. Following a description of the prototype tool, we discuss how and where the prototype was used and the results of those activities.

The Prototype Tool. The prototype is organized around 3 main user functions: part motion path planning, sequence planning and generating technical publications content. All three functions use the concept of an active workspace of parts. The part motion path planning and sequence planning functions work from: a standard workstation setup (computer, monitor and mouse), a HMD and haptic immersive set up, or a monitor haptic setup. The task generation capability expects a standard workstation setup. The prototype has four major styles of user interfaces: a path planning user interface, a procedure planning user interface, a task generation reviewing user interface, and a domain information editing user interface. (The domain information editing user interface was necessary to allow the user to assess the technology.) All three user interfaces are intended as notional interfaces (they are not intended to be robust, commercial grade interfaces). They are intended to show how to use the technology and how the three technical areas integrate with each other.

The path planning interface allows the user to select a part and drag it around in the virtual environment. As the part is dragged, the user defines the part motion path, by 'dropping' path nodes (with a button click). If the haptic device is used, then the haptic device will prohibit the user from defining a path that contains collisions with other parts. If the mouse is used, collisions are reported by coloring the part red when it collides with another part. The paths can be saved out to the file system for future reuse, and an avi can be generated to show the animation of the path.

The sequence planning interface allows the user to manually define a sequence, and/or automatically define a sequence. The sequence is made up of a series of steps. Each step contains one or more parts to be removed/installed. The sequence uses the concept of an initial step; these are all the parts that are to be visible at the start of the sequence.

To manually define a sequence, the user simply selects the next part that is to come off in the next step, and optionally provide a path for this part. The user can either pro-actively add parts to the sequence as individual steps, or as the user picks one part after the other, the parts are automatically added to the sequence as individual steps. The user can jump into path editing mode at any time, and he/she can cut/merge/paste steps in the sequence.

Each part in a step is either being removed or installed (the sequence can have a combination of part removals and installations). If a part is being removed, then when the step is being played, the animation will initially show the part at the beginning of the path, the part will move through the defined path and when the path animation is complete, the part will disappear. Animations of part installations are the same as part removals, but the part remains visible at the end of the path animation.

The user can also 'reverse' the sequence. This is the easiest way to define an assembly sequence. Define the disassembly sequence first and then reverse it. Reversing the sequence reverse the order of and paths in

all the steps but the beginning step. In addition, all parts that were visible in the beginning step and were removed during the body of the sequence (they will be installed in the reversed sequence) are removed from the initial step. Any part that was installed during the sequence (it will now be removed in the reversed sequence) will be made visible in the first step.

As with the path planning, an avi of the complete sequence can be generated and saved to the file system. In addition, the user can load a sequence and kick off 3 versions of the task generation capability:

1. a TG run that regenerates all of the sequence imagery and retrieves the domain information from the database and extends the sequence TGLF by filling in the domain information tags. This run passes the TGLF to the stylesheet transformation that generates a viewable version of the manual content that includes domain information add/delete buttons to allow the user to modify the state of the database domain information. Modifying the domain information database via this presentation of the information will automatically cause the TGLF to be repopulated and the transformation rerun to have the presentation pick up the updates,
2. a TG run that is the same as 1, but bypasses generating the sequence imagery (it assumes that the jpg images and image hot-spot maps already exist), and finally
3. a TG run that uses the already populated TGLF and passes it through another style sheet to present the information, but not support domain information editing.

The Actual Demonstrations. Our demonstration of the technology consisted of using the prototype in a number of ways on two GE Transportation engine programs:

- JSF F136 GE Transportation propulsion unit: We used the prototype to perform a maintainability analysis and generate material for a Critical Design Review (CDR) report out to Lockheed Martin. The maintainability study investigated the removal of the FADEC unit. An avi was generated to show the duct work that needed to be removed to gain access to the unit, and to show how tight the clearance was for removing the FADEC through the access panel.

  We also used the haptic and immersive environment to informally look at the access to the starter generator component. Though no formal material was produced with this analysis, it was informally reported back to Lockheed Martin that the access was exceedingly tight. Subsequent designs showed an enlarged access panel.

- GE Transportation's GP7200 Engine Program: We used the prototype to develop and document assembly procedures for installing the piping on the program's second test build up of an engine. We developed 6 sets of assembly procedures:
  - Core Module Piping Assembly,
  - N Flange Bracket Assembly,
  - Mid Flange Bracket Assembly,
  - E Flange Bracket Assembly,
  - K Flange Bracket Assembly and
  - J Flange Bracket Assembly.

  This exercised included a mix of automated sequence generation and manual sequence generation. In all cases the technicians on the assembly floor indicated that they would prefer to refer to the content generated via the demonstration than the drawings that they were currently using.

  The following table summarizes the results of the sequence generation exercises:

Assembly	Total number of parts	Time to develop the sequence	Manual (M) vs Automatic (A) Sequencing	Estimated time using commercial sequence animation tool
K Flange Bracket	89	1 day	M	3+ days
Core Module Piping	145	1 day	M	10+ days
Core Module Piping	145	4 hours	A	10+ days
N Flange Bracket	70	1 hour	A	3+ days
Mid Flange Bracket	24	1 hour	A	3+ days
E Flange Bracket	100	1 hour	A	3+ days

With each of the sequences listed above the time to develop the sequence strictly speaks to the time to define/generate the order for assembling the parts, plus the time to define/generate the part insertion paths and the time to define the appropriate camera angles on the play back to achieve an acceptable animation. The time figures do not include the time to compile and load the domain information into the database, or the time to actually run the task generation to create the prototype web documentation of the sequence.

The concept is that when the technology is in use, the majority of the domain information will already exist in the database and it should only take < 1 hour to update the domain information for a particular sequence (time to enter the updates).

The domain information that was loaded into the database for the sequences consisted primarily of:

- Torque values for the bolts and nuts
- Lubricants for the nuts
- Bolt hole notes for the various brackets
- A few standard notes about fastener positioning

On average it took < 4 hours to enter the domain information for each sequence.

The task generation took anywhere from 20 minutes to 2 hours depending on the length of the sequence and the number of parts involved. We do not include this in the time figures, since this is a batch process that does not require manual intervention.

The sequences are listed in the order that we developed them. We initially decided not to use the automatic sequencing because we did not expect these parts to be a good candidate set of parts for the algorithm. To use the automatic sequencing, we would have generated full explosion sequences and then reverse the sequence. We realized that though the algorithm would generate a valid removal sequence, it would not be realistic in that it would not properly group the removal of the nuts, bolts, brackets and piping. We anticipated that it would remove all of the nuts, from all around the assembly, then remove the bolts and finally removed the brackets and piping. Instead we were looking to generate sequences where a target group of nuts, bolts and bracket(s) were removed such that the nuts and bolts were the nuts and bolts that held the target bracket(s) in place.

As such we developed our first 3 sequences manually. It turned out that the lions share of the effort was in defining the removal paths and tweaking each paths' play back time so that we could group the nut, bolt and bracket removal all in one step and have the paths in each step play such that after reversing the sequence, the bracket would 'fly' into place first, followed by the bolt and lastly by the nut. There was a learning curve involved here, but by the 3rd sequence we developed a general timing template and the best way to use the demonstration application, to define the appropriate path play back times quickly and efficiently.

After we had the first 3 sequences in place, we experimented with using the sequence generator. With this approach we anticipated we would have to run the sequence generator, then edit the sequence by merging the steps to achieve the appropriate nut/bolt/bracket grouping and adjusting the play back times. We were pleased to find out that using the sequence generation was by far better than the manual technique, even though it still required a fair amount of post editing. The time saved by having the sequence generator define the removal paths, made it more efficient to use the algorithm.

An additional factor that impacted the timing results, was the size of the workspace of parts. We initially started with workspaces that contained all the parts we wanted to have visible in the final documentation. Unfortunately, for the piping sequences this amounted to over 800 parts. We worked with this large data set when we developed the sequence manually, though its size did slow down the step editing functions (adding, merging and deleting). (This slow down was due to the prototype nature of the software and not due to any algorithmic shortcomings.)

The automatic sequencing would not run on the full 800 part workspace (required too much memory). Instead, we learned to pare the workspace down to the critical parts, generated the sequence from this smaller workspace, and then copy the sequence to a workspace with more parts

for generating the documentation. For the N, Mid and E flange bracket assemblies we were able to generate the sequences with the full set of parts (for documentation purposes) present.

The generated content was shown to the craftsmen assembling the test engine. They all indicated that they would prefer to work with the content generated by the SMG program.

4.0 The Next Steps

Our demonstration showed that the SMG technology will make it cost effective and more reliable to develop and document assembly and maintenance procedures directly from the 3D CAD data when compared to developing the traditional 2D line drawing based documentation. In addition, GE Transportation realized that with the digitization of the sequences there are many more opportunities to use this information to improve the product design and delivery. (e.g., fleet level design change impact analysis, service event information capture, etc.)

At this point in the program, we are categorizing the growth opportunities for this technology into three major groups: commercial transition, addition basic research along the original VV/SG/TG lines and new opportunities for improving products and business processes that can be derived from this program.

4.1 Commercial Transition

GE Transportation is developing a technology transition plan for moving the SMG technology out of the research arena and into commercial use. In the short term (spring of 2004), GE Transportation and GE Global Research will perform one more demonstration of the technology on the piping assembly for the JSF F136 engine. By mid 2004 our goal is to develop a strategy for merging the SMG technologies and the GE Transportation military and commercial engine design/build/deploy/support lifecycle. With this strategy in hand we can finalize the application requirements and plan for moving the technology into a commercially supported environment.

Our demonstration activities validated that the basic capabilities are mature enough to start transitioning into the business operations by inserting them in a fully supported application environment. Of course this will take additional resources and effort, but the results of the demonstration show that the benefits that could be achieved with the use of this technology will make it cost effective to transition the current technology to a production worthy state. Since the project is a research oriented project, we did not put much effort into detailing the effort to transition the technology to an application environment. The items listed below are the obvious areas where the transition effort should focus.

- Integration with the Product Data Management system (PDM system) housing the CAD and domain information data
- Adjustments to the business processes to maximize the impact of the technology
- Integrate the user interface with the commercial CAD design/visualization tools

GE will expand and refine this list with the commercialization planning as described in the last paragraph. The cost for productizing the technology will be dependent on how the effort and end product is designed. (e.g., a tight integration with an existing CAD tool will cost more than a loosely-coupled integration.)

4.2 Extend VV/SG/TG Capabilities

With regard to the continued research opportunities, though the technology is at a state where it would be useful in the business operations, it is by no means complete. The following paragraphs outline the major growth areas for each of the 3 technology areas.

Virtual Validation/Immersive Environment. The list below outlines the areas where we recommend further work on the immersive environment technology. It is not meant to be an exhaustive list, nor a prioritized list. We see the virtual validation expanding its focus to include a general improved immersive environment rather than strictly focusing on virtual validation. As such the list below contains development areas that may not necessarily be needed for virtual validation.

Emulate Kinematic Effects in Virtual Environment. At this point in the program we think of this task as representing 2 main capabilities: the ability to simulate the use of tools in the virtual environment, and the ability to represent gravity.

Tooling: As mentioned above, we were not able to implement a tooling solution within the scope of the program, but we were able to design an approach to the solution. This approach is documented here.

We recommend that the tooling effort would best be tackled with a 2 staged effort. The first stage would focus on adding the ability to simulate simple tools (screwdrivers, wrenches, hammers, wedges, etc.) in the environment. The main goal of simulating the use of these tools in the environment would be to assess clearances and to generate documentation. (The goal is not to turn a nut and watch it screw down onto a bolt.) Our first thoughts on the solution were to provide 3D CAD models for the standard tools, plus additional information including a definition of a standard 'connection' for each tool and a standard motion for each tool and the resulting motion for the connected part. The connection definition would identify the areas of the tool that needed to mate with the target part. The motion definitions would then define how the tool and connected part move with respect to each other. The thought was, that when the tool was placed in the 3D environment, the software would automatically determine which part the tool was connected to, then as the tool was moved, the connected part would also move. We recognized that with the faceted approximation to the tool, recognizing the correct connection (100% of the time) would be a difficult task. As such we would develop a semi-automated connection approach whereby the software attempts to automatically discover the tooling connection, but also allows the user to specify the connection. Our design to the first stage of the tooling capability is outlined below.

- For each tool develop a data structure capable of handling the following information:
    - a 3D model for the tool,
    - the polygons on the tools that 'connect' the tool to other parts (tools also)
    - a standard motion for the tool
    - a standard motion (relative to the position of the tool) for the connected parts.
- Develop an interface to allow the user to maintain a toolbox of tools (edit the information above):
    - Define/import a 3D model for the tool
    - Select the polygons on the model that 'connect' the tool to parts
    - Define a standard motion for the tool
    - Define a standard motion (relative to the tool) for the connected parts
- Develop an interface to allow the user to select a tool from the toolbox and insert it into the environment
    - Select a tool from a tool list and place it in the environment
    - Automatically identify connected parts: Extend the mating surfaces analysis (see sequence generation 2.3.3.1) to compute mating surfaces only with respect to the polygons in the tool that have been identified as having the potential to connect to other parts.
    - Use a part coloring strategy to identify the connected parts and allow the user to modify the connected part designation. (via part picking)
- Implement an algorithm that automatically moves the tool and part as per the standard motion specification. As the tool and part are moved, use the collision detection capability from the VV research to identify when the tool/part motion are blocked. Capture and store the non-collision motion using the standard part path data structure.

The second stage to the tooling effort would be to extend the technology to be able to handle tools with complex kinematics, in particular, jointed tools where one part moves in one motion and another part moves in a different motion. Our approach to adding this capability to the tooling solution would be to develop the ability to break a complex tool into a hierarchy/assembly of sub-tools. Again using the polygon selection capability, a complex tool could be broken down into a hierarchy of subsets of polygons. Each subset of polygons would then be considered an individual tool. At the lowest level would be subsets of polygons that represent 'subtools' as the simple tools described in the first stage. The user would never select these 'subtools', instead they would only select and place the highest level representation of the complex tool. The part connections and tool and part motions would be determined by merging all the effects of the defined 'subtools'.

Gravity: Gravitational effects could be achieved in the virtual environment by extending the part information to carry weight data and adjusting our force feedback computation to factor in 'down' forces that are a function of the part's weight. However, the only reason that we see for simulating gravity is to allow the user to determine if the part is too heavy to be held. The problem with this is that determining what is 'too heavy' is too subjective. It would be better to use the ergonomic analysis tools (such as the

Jack tool offered by EDS[8]) that already exist to assess the ease of moving a part with respect to its weight and with respect to the various size breakdown of the general population.

Develop and Implement Hybrid Collision-Detection Techniques. Our thoughts on this capability would be to extend our part map/point/volume sampling technique to support links back into the original polygonal and/or nurb part definitions. As the point sampling collision analysis reaches certain thresholds of collision, the analysis would further refine the collision detection by analyzing the corresponding areas in the polygonal and/or nurb representation. We anticipate the key developments with this research to be the linking data structures to allow the analysis to quickly move to the desired subset of each representation, and determining the appropriate threshold values that trigger refining, expanding the analysis.

Implement Tracking of Multiple Human Events in Virtual Environment. We are including this extension in our discussion of the basic research growth areas primarily because of the increased computational complexity, which is incurred as we add more devices. Work would need to be done to determine how many devices the various computing environments could support. The advances in the hybrid collision detection capabilities would also heavily impact the results of this analysis.

That said, with our object oriented approach to the implementation, extending the analysis for multiple hands, multiple people, and/or potentially multiple haptic devices would be handled by extending the technology to support multiple tracking objects and multiple haptic objects.

There would be additional work involved with this extension in implementing how these objects should be allowed to interact with each other (though this could be thought of as an application transition growth area). It is the requirements of the underlying application that requires the multiple objects that will determine how the objects interact. For example, if the desire is to support multiple users with HMDs to be viewing the environment, then we probably need to develop a representation of a user and place instances of that geometry in the environment so the users can see each other (mainly to help them avoid running into each other). With multiple hands/haptic devices, the big issue would be how to turn the haptic devices on/off (given that both hands are holding a stylus).

Levels of Reality. With our demonstration activities, we discovered that the level of reality needed in the environment was determined by the different reasons for using the immersive technology. For example, when using the environment to determine if a part could fit through an opening, the units of the haptic device motion did not have to have a 1-1 ratio with the units of the geometry. As another example, we discovered that sometimes it we needed to be able to adjust the view angle on the HMD to adjust the resulting view perspective based on why the HMD was being used. If the HMD was being used to determine if the user could see or reach a part from a certain position, then we needed to have the HMD view angle match reality, if however, the user just wanted to be able to look a design over, then we would often widen the view angle to allow the user a wider view, in essence, making the geometry appear further away than it really was.

We recognize that a great deal of effort could be put into characterizing how the technology could/should be used and then characterizing how 'realistic' the technology needs to be to support these uses.

Sequence Generation. The list below outlines the areas where we recommend further work on the sequence generation technology. It is not meant to be an exhaustive list, nor a prioritized list.

Incremental sequence generation. Though we did extend the sequence generation to support iterative sequence generation, we recommend more effort be put into evaluating the effectiveness of the algorithm. In addition, as we mentioned above, the incremental sequencing that we implemented enforces a no collision rule, in that it will alter the input sequence if necessary. We recognize that sometimes it may be the case that the user wants the input sequence to be preserved even if the analysis determines that this would cause collisions. As such the incremental algorithm should be extended to allow the user to specify whether the input sequence can be reordered, and the algorithm, should then try to minimize collisions if they are unavoidable with the initial sequence.

Heuristics for grouping part removal. We did initially implement the ability to recognize when parts needed to be grouped and removed as a unit so as to avoid leaving parts 'floating' in space. We decided to pull this out of the algorithm because, after testing it out on numerous real-life designs we found that it

tended to over-group parts mainly because of the imprecise nature of the design data (missing part definitions, faceted approximations, etc.).

We feel that improvements in this area will come more from the development of heuristics that use a general characterization of the part type (nut, bolt, bracket, o ring, etc.) rather than an analysis of the geometry. For example, learning from our GP7200 exercise, it would be nice if the sequencer would try to order steps so that predefined groups of part types that are physically close together come off in steps that are also close to each other (e.g., the sequencer could try to generate sequential steps to remove nuts, bolts and brackets that are physically close together).

Multi-linear part removal. As mentioned above, our sequencing algorithm is limited in that it only computes linear part removal paths. The discussion of the multidirectional explosion task in the Sequence Generation section covers our work-around to this shortcoming (using our semi-automated path planning capability). We would recommend that an investigation of multi-linear part removal start with this approach and focus on automatically determining a 'general' removal direction as an initial guide path. Perhaps results from the part grouping heuristics could help determine this guide path.

Sequence validation. We recognize the need to 'plug' a sequence into a new workspace, let the user edit a few steps (mainly to swap an old part for a new design) and then run a process that reports out on whether the sequence is valid or not. This could be thought of as an extension of the incremental sequencing, it simply needs to be able to preserve the original sequence without making any changes, and report out on the validity of the sequence.

Path line animations. This extension is covered in the discussion about developing understandable mating lines for exploded parts in the Sequence Generation section.

Automatic camera placement. As the sequence is being generated it would be a considerable timesaving if the algorithm also generated an initial camera position for each step.

Task Generation. The main growth area we see for this technology is to bring it out of the information delivery arena and into the information collection arena. That is to use the basic infrastructure and extend the xml transformation capability so that instead of generating technical manual content, it generates form content that contains the technical manual content and data collection points to record information about the procedure as it is performed. For instance, if a step calls for the installation of a serialized part, the content would contain the instructions for installing the part, plus the hooks for collecting the installed part's serial number.

4.3 Other Opportunities Derived from this Program.

All along we have claimed that large product manufacturing can benefit greatly if they can have a cost effective mechanism for developing and maintaining a fine-grained digital record of the product procedures. With this program we set out to develop techniques to make it cost effective to develop these digitized procedure at the granularity we envisioned and to demonstrate the initial obvious use of this information; the generation of technical manual content.

With the successful demonstration of the technology, we have been able to identify numerous additional opportunities for dramatically improving product development and product support. The following list highlights the most visible opportunities. We are confident that more will be discovered as we continue to realize the full impact of the technology.

Maintenance Analysis, Task Time/Complexity Analysis. GE GRC has already embarked on follow-up project that uses the digitized sequences and a database of standard task times to automate estimating the time to perform a procedure. This information is captured and rolled into the product's Lifecycle Support Analysis (LSA).

Fleet Level Change Impact Analysis. With the digitized procedures, design changes can be evaluated based on their impact on the deployed fleet's maintenance procedures. When a part change is proposed, extensions to this program's technology could identify all the sequences that are affected by the change, and the impact of the change. This analysis could then be coupled with a fleet-wide analysis to automatically assess the fleet level impact of the change.

Autonomic Logistics –the Right Information at the Right Time. The reduced cost to generate manual content will move the industry to maintaining manuals that are tailored to each product configurations (rather than having one manual apply to multiple configurations). With this, autonomic logistics tools can deliver instructions tailored to the user's needs.

Automated Workscope Merging. The sequence generation capability could be extended to be able to merge multiple procedures. This could form the basis for a tool that automatically develops a single workscope from several individual workscopes.

5.0 Conclusion

We had started the program with a well-defined technical challenge and what we thought was a relatively well understood user community. As we progressed with the research and discovered solutions to the technical challenges we also discovered that the technologies being developed would have a far broader impact on the aircraft engine design, build, and deploy process than originally anticipated. We have implemented our initial solutions to the technical challenges and demonstrated the effectiveness of these solutions to the expanded user community.

- We demonstrated the technology on the GE Transportation F136 (JSF) engine/airframe interface design to perform a maintainability study on the FADEC and starter generator components.
- GE Transportation used the demonstration technology to develop the test build assembly instructions for the prototype builds of their new commercial GP7200 engine. Not only was the technology used to generate and validate the instructions, but the content generated was delivered to the assembly floor in lieu of their standard instruction packet.

As a result GE Transportation is actively involved in developing a transition plan to move the technology from its current demonstration state to a supported, commercial tool.

6.0 Acknowledgements

This technology was conceived and developed by a team of very talented individuals. I would like to acknowledge their participation in the effort and thank them for their contributions: Dr. Russell Blue, Richard Chadwick, Mike Dometita, Dr. Louis Hoebel, Anne Kelly, Steve Linthicum, Ken Murphy, Pascale Rondot, George Ryon, Craig Silber, Dr. Christopher Volpe, Bruce Wilde, Bowden Wise, Dr. Boris Yamrom. We would also like to thank Jeff Wampler from the AFRL/HEAL for his invaluable input.

7.0 Publications

1. Wampler, J.L., Bruno, J.M., Blue, R.S., Hoebel, L.J., 2003 (January), *Integrating Maintainability and Data Development*, The Annual Reliability and Maintainability Symposium.
2. Bruno, J.M., Blue, R.S., Wampler, J.L., *Achieving Reuse of Procedural Specifications*, 2003 (October), Canadian Reliability and Maintainability Symposium
3. Volpe, C., and Blue, R.S., *Virtual haptic validation for service manual generation*, Proceedings of Phantom User's Group, 2002.

8.0 References

1 The Visualization Toolkit (VTK). http://www.vtk.org, April, 2004.

2 Active Tcl/Tk, http://tcl.activestate.com, April 2004.

3 MYSQL, http://www.mysql.com, April 2004.

4 W.A. McNeely, K.D. Puterbaugh, J.J Troy. "Six Degree of-Freedom Haptic Rendering Using Voxel Sampling." Proceedings ACM SIGGRAPH 99 Conference, August 1999, Los Angeles, CA, pp. 401-408, 1999.

5 SensAble Technologies, Inc. ,15 Constitution Way, Woburn, MA 01801,Tel: +1 (781) 937-8315, http://www.sensable.com/index.asp, April 2004.

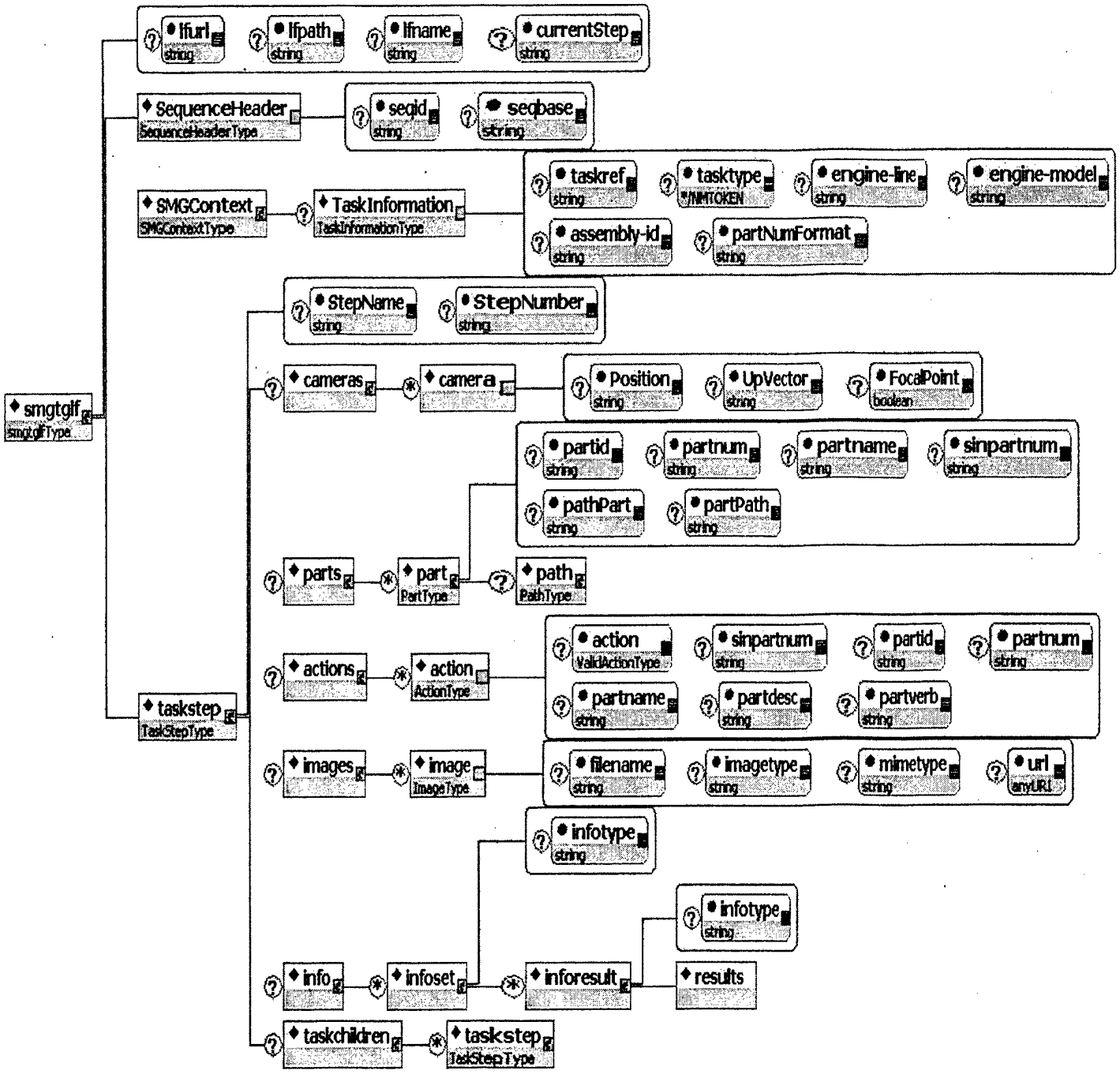
6 Ascension Technology Corporation, P.O. Box 527 Burlington, VT 05402, USA, (802) 893-6657, http://www.ascension-tech.com/products , April 2004.

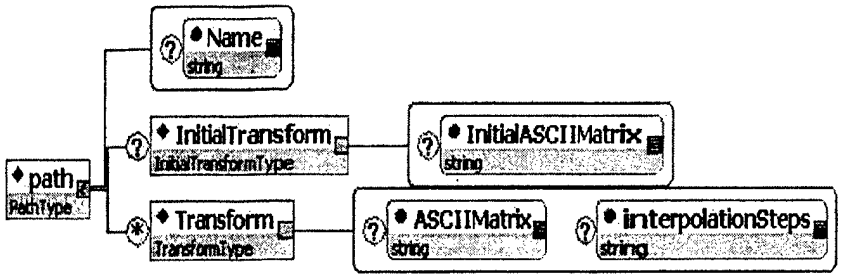
7 Immersion, CyberGlove, 801 Fox Lane, San Jose, CA 95131 USA, 408.467.1900, http://www.vrealities.com/cyber.html, April 2004.

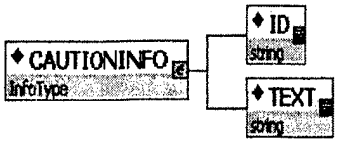
8 EDS, http://www.eds.com/, April 2004.

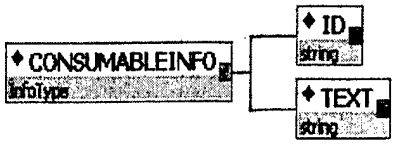
Appendix A – Natural Language Generation Logical Form

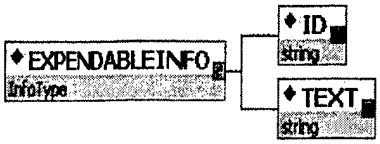
The following diagram illustrates the Logical Form. Below the diagram is a description of each entry.

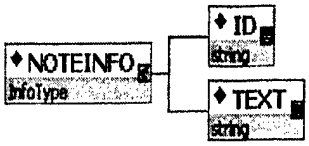


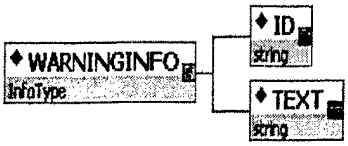


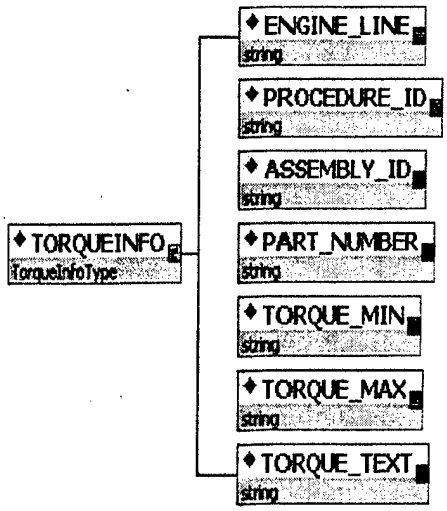


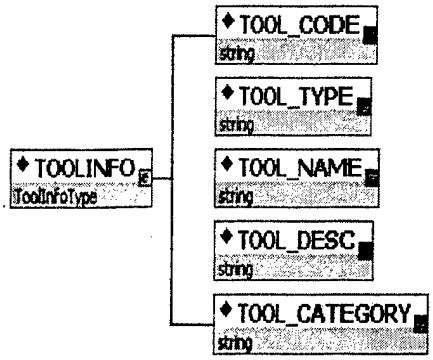












The logical form is described as follows:

Smgtglf – the top level tag.

Attributes:

lrurl – the logical form file name
lfpath – the path to the logical form file
lfname – the name of the sequence
currentStep - the step to display first when the html is generated.

Inner tags:

SequenceHeader – header information about the sequence
SMGContext – context information about the sequence
Taskstep – a hierarchical list of sequence steps. An implementation convention established a single top level taskstep. The actual sequence steps are created as substeps within this top level taskstep.

SequenceHeader – header information about the sequence.

Attributes:

Seqid – the identifier for the sequence – for the prototype – the sequence name
Seqbase – not used

Inner tags: none

SMGContext – context information about the sequence

Attributes: none

Inner tags:

TaskInformation – information about the task

TaskInformation

Attributes:

taskref – the task identifier
tasktype – remove or install (used as a default action in the query expansion)
engine-line – the engine line the sequence is for (used in the query expansion)
engine-model – the engine model the sequence is for (not used)
assembly-id – the assembly the sequence is for (used in the query expansion)
partNumFormat – a key controlling how to split the CAD filename into part number, part instance, and ppn number

Inner tags: none

Taskstep – a sequence step. There may be multiple tasksteps within the smgtglf tag, and an individual taskstep may contain a subsequence within it.

Attributes:

StepNumber – the step number
StepName – the step name

Inner tags:

Cameras – a list of the camera positions to move the camera through before the step paths are played
Parts – a list of the parts active in this step
Actions – a list of the actions on each part in the step
Images – an ordered list of the images that comprise the animation of the step
Info – the domain information for this step
TaskChildren – a subsequence for the step

Cameras – a list of the camera positions to move the camera through before the step paths are played

Attributes: none

Inner tag:

Camera – a list of the information about individual camera positions.

Camera – the information about an individual camera position

Attributes:

Position – the position of the camera
UpVector –the vector indicating the up position for the camera
FocalPoint – false (a Boolean value used by the camera handling)

Inner tag: none

Parts – a list of the parts active in this step

Attributes: none

Inner tag:

Part – a list of the information about individual parts in the step.

Part – the information about an individual part in the step

Attributes:

PartId – the full part id
PartNum – the part number from the part id
PartName – the part name
SinPartNum – the PPN(SIN) number from the part id
PartPath – the path to the CAD part file

Inner tag:

Path – the path defining the part motion in the sequence.

Path – the path defining the part motion in the sequence.

Attributes:

PathName – the name of the path

Inner tag:

InitialTransform –The beginning position for the part.

Transform – the list of transforms that define the path.

InitialTransform – a transformation matrix that will move the part to the beginning of the path.

Attributes:

InitialASCIIMatrix – the transformation matrix (in ascii format) for positioning the part at the beginning of the path.

Inner tag: none

Transform – a transformation matrix that will move the part to the next node in the path.

Attributes:

ASCIIMatrix – the transformation matrix (in ascii format) for positioning the part at the next node in the path.

InterpolationSteps – the number of intermediate interpolations to move the part through before finally moving the part to the position defined by the ASCIIMatrix.

Inner tag: none

Actions – a list of the actions on each part in the step

Attributes: none

Inner tag:

Action – a list of the actions on the individual parts in the step.

Action – the information about an action on an individual part in the step

Attributes:

Action – one of install, remove,
SinPartNum – the PPN(SIN) number from the part id
PartId – the full part id
PartNum – the part number from the part id
PartName – the part name
PartDesc – the description of the part
PartVerb – the verb for the action

Inner tag: none

Images – an ordered list of the images that comprise the animation of the step

Attributes: none

Inner tag:

Image – a list of the images that comprise the animation of the step.

Image – the information about an individual image in the step animation

Attributes:

FileName – the filename for the image
ImageType – the type of the image (jpg)
MimeType – the mime type for the image
URL – the url for the image

Inner tag: none

Info – the domain information for this step
Attributes: none
Inner tag:
InfoSet – a list of the chunks of domain information that correspond with this sequence step.

InfoSet - a list of the chunks of domain information of a particular information type that correspond with a particular part in the sequence step

Attributes:
InfoType – the type of domain information: warning, note, caution, expendable, consumable tool, or torque
Inner tag:
InfoResult – a list of the pieces of a particular information type that correspond with a particular part in the sequence step

InfoResult – a piece of domain information of the appropriate type that correspond with this sequence step.
Attributes:
InfoType – the type of information (warning, note, caution, expendable, consumable tool, or torque)

Inner tag:
Params – a list of the parameters used in the original call to retrieve the domain information
Results – a list of the results of the retrieval

Params - a list of the parameters used in the original call to retrieve the domain information
Attributes: none
Inner tag:
Param – a parameter value passed to the retriever

Param - a parameter value passed to the retriever
Attributes:
Name: the name of the parameter (engine line, procedure id, assembly id, or part number)
Value: the value of the parameter
Inner tag: none

Results – a list of the results of the retrieval
Attributes: none
Inner tag:
CautionInfo – a caution from the domain information database
ConsumableInfo – a consumable from the domain information database
ExpendableInfo – an expendablefrom the domain information database
ToolInfo – a tool from the domain information database
NoteInfo – a note from the domain information database
TorqueInfo – a torque from the domain information database

WarningInfo – a warning from the domain information database

CautionInfo – a caution from the domain information database
Attributes: none
Inner tag:
ID – the primary index value from the domain information database (used to merge multiple retrievals of the same piece of domain information into a single presentation of this information).
Text – the domain information

ConsumableInfo – a consumable from the domain information database
Attributes: none
Inner tag:
ID – the primary index value from the domain information database (used to merge multiple retrievals of the same piece of domain information into a single presentation of this information).
Text – the domain information

ExpendableInfo – an expendablefrom the domain information database
Attributes: none
Inner tag:
ID – the primary index value from the domain information database (used to merge multiple retrievals of the same piece of domain information into a single presentation of this information).
Text – the domain information

NoteInfo – a note from the domain information database
Attributes: none
Inner tag:
ID – the primary index value from the domain information database (used to merge multiple retrievals of the same piece of domain information into a single presentation of this information).
Text – the domain information

NoteInfo – a note from the domain information database
Attributes: none
Inner tag:
ID – the primary index value from the domain information database (used to merge multiple retrievals of the same piece of domain information into a single presentation of this information).
Text – the domain information

ToolInfo – a note from the domain information database
Attributes:
Code –the tool code
Type – the tool type
Desc – the tool description

Name – the tool name
Category – the tool category (standard, local, special)
Inner tag: none

TorqueInfo – a torque from the domain information database
Attributes:
EngineLine – the engine line
Procedure – the procedure id
Assembly – the assembly id
PartNumber – the part number
Text – the torque text
Inner tag: none

TaskChildren – a subsequence for the step
Attributes: none
Inner tag:
TaskStep

It is important to note that this description of the logical form is the critical subset of the schema that was implemented for the SMG project. The additional tag entries in the implemented xsd allowed the digitized sequence to be reused in a Maintenance Time, Task Time roll-up tool that was developed under an internal GE project. This additional use of the sequence underscores the power that can be gained from the combination of the xml approach and having the procedure information maintained at a fine grained digitized level.

Appendix B – Domain Information Characterization

As a result of analyzing the existing manuals we divided the domain information into information sets and defined the indexes that were needed to find the domain information. The following describes each information set and the indexes.

Warnings – all the warnings that may appear in technical manuals. For each warning, indexes are maintained to associate the warning with the product in the following ways:

- The warning may be general to all documentation for the engine. As such simply the engine line will index it.
- In addition, the warning may apply to all documentation concerning a given assembly. As such the assembly name and engine line may index it.
- In addition, the warning may apply to all steps in a given procedure. As such the procedure name, assembly, and engine line may index it.
- In addition the warning may be specific to a part and or part instance. As such it may also be indexed by part and/or part instance.
- In addition the warning may be specific to an action on a part (remove, install, etc.). As such it may also be indexed by the action.

Indices: In summary, the warnings are indexed by any combination of engine, assembly, procedure, part/part instance. Each warning must be indexed by at least one of the items from the previous list. The warning may be further restricted to apply to a specific action.

Query Expansion: When the task generation retrieves warnings the retriever is called with the prioritized indices: engine number, assembly name, procedure name, part. In addition action as provided an absolute index.

If a part number or part instance identifier is supplied then the query expansion mechanism will look for all warnings that are indexed by the part and all subsets of engine number, assembly name, procedure name and action. If no part number is supplied then the query expansion mechanism will look for all warnings that are indexed by all subsets of the engine number, assembly name and procedure name.

Notes – all the notes that may appear in technical manuals. For each note, indexes are maintained to associate the note with the product in the following ways:

- The note may be general to all documentation for the engine. As such it will be indexed simply by the engine number.
- In addition, the note may apply to all documentation concerning a given assembly. As such it may also be indexed by the assembly name.
- In addition, the note may apply to all steps in a given procedure. As such it may also be indexed by the procedure name.
- In addition the note may be specific to a part and or part instance. As such it may also be indexed by part and/or part instance.
- In addition the note may be specific to an action on a part (remove, install, etc.). As such it may also be indexed by the action.

Indices: In summary, the notes are indexed by any combination of engine, assembly, procedure, part/part instance. A value must be provided for one of the items from the previous list. The note may be further restricted to apply to a specific action.

Query Expansion: When the task generation retrieves notes the retriever is called with any combination of engine number, assembly name, procedure name, part, part instance and action. If a part number or part instance identifier is supplied then the query expansion mechanism will look for all notes that are indexed by the part and all subsets of engine number, assembly name, procedure name and action. If no part number is supplied then the query expansion mechanism will look for all notes that are indexed by all subsets of the engine number, assembly name and procedure name.

Cautions– all the cautions that may appear in technical manuals. For each caution, indexes are maintained to associate the caution with the product in the following ways:

- The caution may be general to all documentation for the engine. As such it will be indexed simply by the engine number.
- In addition, the caution may apply to all documentation concerning a given assembly. As such it may also be indexed by the assembly name.
- In addition, the caution may apply to all steps in a given procedure. As such it may also be indexed by the procedure name.
- In addition the caution may be specific to a part and or part instance. As such it may also be indexed by part and/or part instance.
- In addition the caution may be specific to an action on a part (remove, install, etc.). As such it may also be indexed by the action.

Indices: In summary, the cautions are indexed by any combination of engine, assembly, procedure, part/part instance. A value must be provided for one of the items from the previous list. The caution may be further restricted to apply to a specific action.

Query Expansion: When the task generation retrieves cautions the retriever is called with any combination of engine number, assembly name, procedure name, part, part instance and action. If a part number or part instance identifier is supplied then the query expansion mechanism will look for all cautions that are indexed by the part and all subsets of engine number, assembly name, procedure name and action. If no part number is supplied then the query expansion mechanism will look for all cautions that are indexed by all subsets of the engine number, assembly name and procedure name.

Expendables – all the expendables that may appear in technical manuals. For each expendable, indexes are maintained to associate the expendable with the product in the following ways:

- The expendable may be general to all documentation for the engine. As such it will be indexed simply by the engine number.
- In addition, the expendable may apply to all documentation concerning a given assembly. As such it may also be indexed by the assembly name.
- In addition, the expendable may apply to all steps in a given procedure. As such it may also be indexed by the procedure name.
- In addition the expendable may be specific to a part and or part instance. As such it may also be indexed by part and/or part instance.
- In addition the expendable may be specific to an action on a part (remove, install, etc.). As such it may also be indexed by the action.

Indices: In summary, the expendables are indexed by any combination of engine, assembly, procedure, part/part instance. A value must be provided for one of the items from the previous list. The expendable may be further restricted to apply to a specific action.

Query Expansion: When the task generation retrieves expendables the retriever is called with any combination of engine number, assembly name, procedure name, part, part instance and action. If a part number or part instance identifier is supplied then the query expansion mechanism will look for all expendables that are indexed by the part and all subsets of engine number, assembly name, procedure name and action. If no part number is supplied then the query expansion mechanism will look for all expendables that are indexed by all subsets of the engine number, assembly name and procedure name.

Consumables – all the consumables that may appear in technical manuals. For each consumable, indexes are maintained to associate the consumable with the product in the following ways:

- The consumable may be general to all documentation for the engine. As such it will be indexed simply by the engine number.
- In addition, the consumable may apply to all documentation concerning a given assembly. As such it may also be indexed by the assembly name.
- In addition, the consumable may apply to all steps in a given procedure. As such it may also be indexed by the procedure name.

- In addition the consumable may be specific to a part and or part instance. As such it may also be indexed by part and/or part instance.
- In addition the consumable may be specific to an action on a part (remove, install, etc.). As such it may also be indexed by the action.

Indices: In summary, the consumables are indexed by any combination of engine, assembly, procedure, part/part instance. A value must be provided for one of the items from the previous list. The consumable may be further restricted to apply to a specific action.

Query Expansion: When the task generation retrieves consumables the retriever is called with any combination of engine number, assembly name, procedure name, part, part instance and action. If a part number or part instance identifier is supplied then the query expansion mechanism will look for all consumables that are indexed by the part and all subsets of engine number, assembly name, procedure name and action. If no part number is supplied then the query expansion mechanism will look for all consumables that are indexed by all subsets of the engine number, assembly name and procedure name.

Special Tools – all the special tools that may appear in technical manuals. For each special tool, indexes are maintained to associate the special tool with the product in the following ways:

- A special tool is specific to a part and or part instance. As such it is indexed by part and/or part instance.
- In addition, the special tool may apply only to instances of the part when it is in a given assembly. As such it may also be indexed by the assembly name.
- In addition the special tool may be specific to an action on a part (remove, install, etc.). As such it may also be indexed by the action.

Indices: In summary, the special tools are indexed by one of the following index combinations part, part instance, part/assembly, or part instance/assembly. The special tool may be further restricted to apply to a specific action.

Query Expansion: The retriever for the tooling information looks for tools that are specific to the supplied part/part instance, action, and assembly index value. There is no expansion of this query.

Standard Tools – all the standard tools that may appear in technical manuals. For each standard tool, indexes are maintained to associate the standard tool with the product in the following ways:

- A standard tool is specific to a part and or part instance. As such it is indexed by part and/or part instance.
- In addition, the standard tool may apply only to instances of the part when it is in a given assembly. As such it may also be indexed by the assembly name.
- In addition the standard tool may be specific to an action on a part (remove, install, etc.). As such it may also be indexed by the action.

Indices: In summary, the standard tools are indexed by one of the following index combinations part, part instance, part/assembly, or part instance/assembly. The standard tool may be further restricted to apply to a specific action.

Query Expansion: The retriever for the tooling information looks for tools that are specific to the supplied part/part instance, action, and assembly index value. There is no expansion of this query.

Local Tools – all the local tools that may appear in technical manuals. For each local tool indexes are maintained to associate the special tool with the product in the following ways:

- A local tool is specific to a part and or part instance. As such it is indexed by part and/or part instance.
- In addition, the local tool may apply only to instances of the part when it is in a given assembly. As such it may also be indexed by the assembly name.
- In addition the local tool may be specific to an action on a part (remove, install, etc.). As such it may also be indexed by the action.p

Indices: In summary, the local tools are indexed by one of the following index combinations part, part instance, part/assembly, or part instance/assembly. The local- tool may be further restricted to apply to a specific action.

Query Expansion: The retriever for the tooling information looks for tools that are specific to the supplied part/part instance, action, and assembly index value. There is no expansion of this query.

Torques – all the torques that may appear in technical manuals. For each torque, indexes are maintained to associate the torque with the product in the following ways:

- The torque is specific to a part and or part instance. As such it must be indexed by part and/or part instance.
- In addition, the torque value may further restricted to a specific procedure. As such it may also be indexed by the procedure name.
- In addition, the torque value may further restricted to a given assembly. As such it may also be indexed by the assembly name.
- And finally, the torque value may be even further restricted to a specific engine. As such it will be indexed simply by the engine number.

Indices: In summary, the torques are indexed by any combination of engine, assembly, procedure, part/part instance. A value must be provided for one of the items from the previous list.

Query Expansion: The query expansion mechanism operates as follows. The retriever for the torque information first looks for torques that are specific to the supplied part instance, procedure, assembly, and engine index value. If no torque is found the retriever looks for a torque that is specific to the part, procedure, assembly and engine index. If still no torque is found the retriever drops the engine value from the query and looks for a torque specific to the part instance and then the part. If still no torque is found, the retriever drops the assembly value from the query and finally the retriever drops the procedure is from the query and looks for a torque specific to the part instance and then the part. As soon as a torque value is found, the retriever ends it search and returns that value.

Appendix C - Demonstration Prototype User Manual

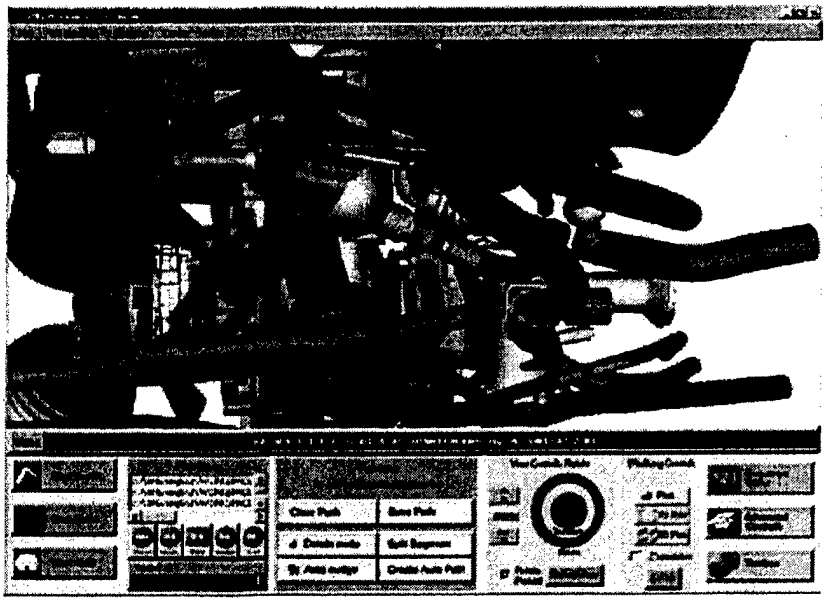
As part of the prototype effort we developed a web based user manual to describe the prototype environment and help the user navigate through it. The user manual consists of a set of stand-alone web pages. They have been delivered with this document.

The following pages contain excerpts from the SMG User Manual.

Overview

This is a User Manual for the Maintainability Tool. It is a web based manual that will help you navigate through topics of interest. The left area presents the Table of Contents from where you can navigate; the topic will be displayed in the center or main area. The bottom area represents the application panel; you can also click on any item of interest on the panel and the relative information will be displayed on the center or main area.

Screen Layout Concept:



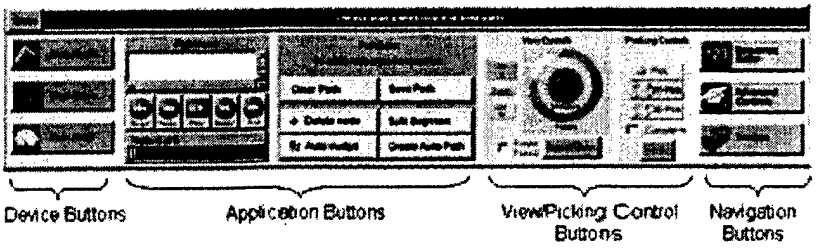
The application window is composed in 3 major areas:

Application Menus - The menu bars at the top of the display will primarily be used to set up the environment. It will provide options and settings as well as Loading and Exit.

Rendering Window - The majority of the display is the rendering window/view. This will contain the 3D CAD representation plus additional geometry for the glove and haptic device, and for the

path/sequence planning activities (paths, haptic sword, etc.) Clicking in that area will let you rotate/translate/zoom the workspace or select some parts.

Application Panels - At the bottom of the display will be the application panels. There will be 3 main panels: path planning/editing, sequence planning/editing, and advanced controls (part viewing, part grouping, etc.). The Application Panel will give you all functionalities you need to perform a typical task (creating a path or creating a sequence).



The application panel is separated in 4 sections:

1 - **Device Buttons:** The device buttons turn the Virtual Reality (VR) devices on and off.

2 - **Application Buttons:** The application buttons support the current activity. There will be two sets of application buttons, path editing buttons and sequence editing buttons, depending on which module you are working.

3 - **View/Picking Control Buttons:** The part view control buttons give the user the ability to interact with the display. This area of the panel allows the user to rotate, translate, zoom in and out and pick node of a path.

4 - **Navigation Buttons:** The navigation buttons provide the means to navigate through the main functions in the environment: path editing, sequence editing, advanced controls, and interacting with the toolbox.

Basic Task Definition:

Part removal (path planning)

The user will be able to define part removal paths. The environment will support both manual and automatic path planning.

- With the manual path planning, the user attaches to a part and drags the part along the desired path. As the part is being dragged the user can drop path nodes to record the path.
- With automatic path planning the user defines the target end point for the part and selects Create Auto Path feature.
- The user will also be able to edit paths: add, remove path nodes, etc.

Sequence definition

The user will be able to define part removal/assembly sequences. The environment will support both manual and automated sequence generation.

- With the manual sequence definition the user will identify the part(s) that come off/on during each step in the sequence. Sequence steps will support the removal of multiple parts in a single step. Each part removal/assembly will have the option of reusing a previously defined path.
- With automatic sequence generation a predefined set of paths and sub-sequences may be provided. The generation algorithm will attempt to discover a sequence that adheres to the input paths and sub-sequences. Sequence generation will be for both full assembly/disassembly and individual part removal.

VR Devices Concept:

The application may work under several different hardware configurations:

- Display and Mouse;
- Display and Haptic Device;
- Display and Glove;
- Helmet Mounted Display (immersive) and Haptic Device;
- Helmet Mounted Display (immersive) and Glove.

Haptic Device

The haptic device is a link between your arm and the virtual environment. The haptic device provides you force feedback while manipulating virtual objects in the virtual environment; this way

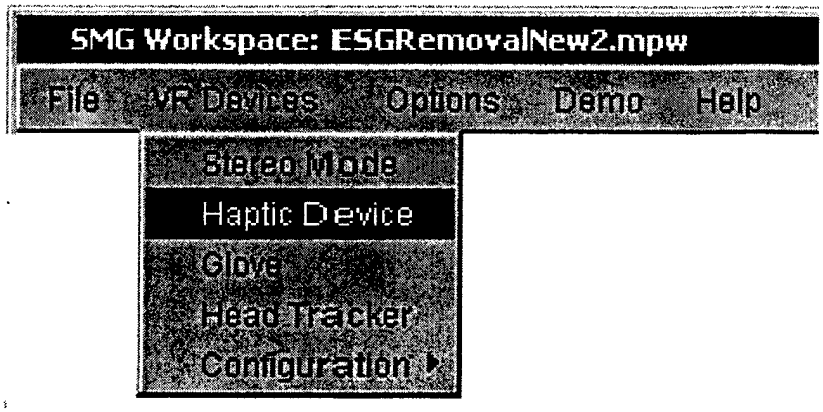
you will be able to feel a collision with another object. The haptic device will be represented as a sword in the rendering area. The end of the sword is the pointer.

Basic actions with the haptic device:

- To pick a part: Click once with the button on the top of the haptic device handle;
- To attach to a part: Click twice with the button on the top of the haptic device handle;
- To drop nodes along a path: Click once after being attached to the part;
- To unattach: Click twice when you are attached.

The haptic device is used as a mouse on the application panel, clicking on the button, on top of the haptic device handle, is the equivalent of left clicking on a normal mouse.

If you are working with a Haptic Device, select VR Devices > Haptic.



Then, you will be able to start the haptic device from the Application Panel whenever you want to use it.

When user clicks on this button the first time, a panel to calibrate the haptic device appears. All the following times it will start or stop the haptic device.

Cyberglove Device

The Cyberglove device allows you to show your hand and its movements in real time in the virtual environment. The Cyberglove doesn't provide you force feedback while manipulating virtual objects in the virtual environment. The button located at the wrist on the Cyberglove may be use as a left mouse button:

Basic actions with the Cyberglove:

- To pick a part: Click once with the button on the wrist of the Cyberglove;
- To attach to a part: Close your fist around the part;
- To drop nodes along a path: Click once after being attached to the part;
- To unattach: Open your fist.

The Cyberglove is used as a mouse on the application panel, clicking on the button on the wrist is the equivalent of left clicking on a normal mouse.

If you are working with a Glove Device, select VR Devices > Glove.

Helmet Mounted Display

If you are working with a Helmet Mounted Display, select VR Devices > HMD.

To start the HMD, click on "Start HMD" on the application panel. From then the movement of your head while wearing the HMD will be tracked and the point of view on the virtual scene will change accordingly. To stop the HMD click on "Stop HMD" on the application panel.

Workspace Concept:

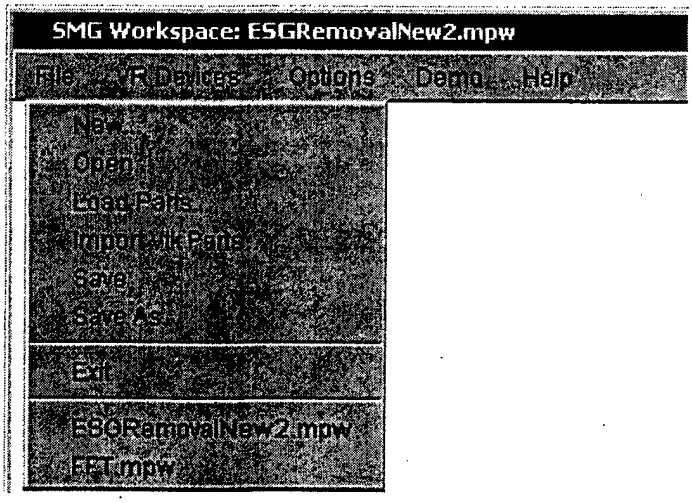
A workspace is a set of data where additional information may have been added. For example a workspace may include haptically moveable parts or collidable parts. If a path has been saved during a part removal, the path will be associated with the workspace used when the path has been saved. Also, a sequence will be associated with the workspace used when the sequence has been created. All workspaces are kept in your central area under VV_PRODUCTION_DATA\WORKSPACES\

Creating a Workspace

To create a workspace you need to load data or to import data. If the data comes from CAD data (UG or CATIA) checked in IMAN you will only need to load the data which will be under your central area under VV_PRODUCTION_DATA\DATA. If you have data that has not been checked in IMAN you will have to first transfer the data into vtk and then import the vtk data. The procedures to load data or to import VTK parts are described below.

Load Parts

To load parts, go in the File Menu and select "Load Parts".



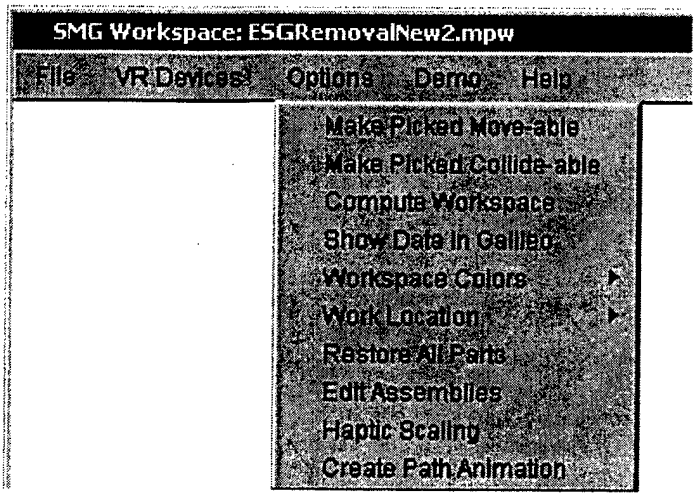
Then select the directory from which you want to load your data and click the "Ok" button. The parts will be shown in the rendering area.

Import Vtk Parts

To import VTK data, go in the File Menu and select "Import Vtk Parts". Then select the directory from which you want to import your data and click the "Ok" button. The parts will be shown in the rendering area.

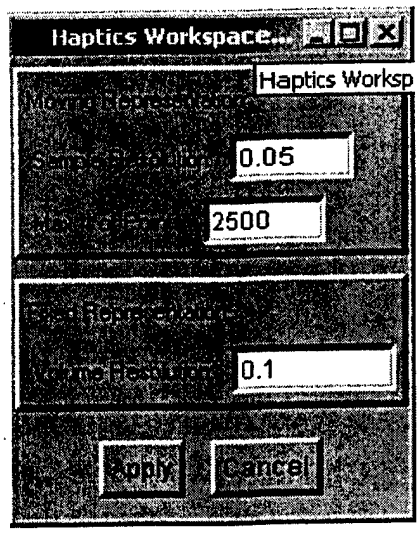
Haptic Workspace

At first, when you create a workspace, there are no haptically moveable parts or collidable parts. To define collidable or haptically moveable parts you have to pick them and then select, on the menu "Options", "Make Picked Move-able" or "Make Picked Collide-able". Then select, on the menu "Options", "Compute Workspace".



Compute Workspace

When you click on "Compute Workspace", the following dialog box will appear:



Parameters definition:

Moving Representation

Sample Resolution: this number indicates the approximate spacing between the sample points on the moving object.

Max # of Points: This setting is used to limit the number of points an object has. If the specified sample resolution gives too many points, then the number of sample points on the object will be approximately equal to the max number. The Sample Resolution should be approximately equal to the volume resolution.

Fixed Representation

Volume Resolution: The Volume Resolution specifies the spacing between points in the volume... the closer the points, the greater the precision of the collision detection, but the greater the time to compute and disk space required.

After entering data in these fields you can click apply and the haptic workspace will be created. You will see the progression bar in the panel title bar giving you an estimate of the time remaining

to complete the task. When the Haptic Workspace is done you should save it. To save the workspace, click on the Menu/ File/ Save.

Part visibility

In any workspace you have the control on the parts visibility. You can choose between solid or wireframe or even make a part invisible. This may be useful to see a hidden part without moving the parts that hide it. To make a part invisible, go on "Advanced Controls" Panel, picked the parts you want the visibility modified and click on "Wireframe/Solid" or "Invisible" depending on what you want to do.

Part color

The parts are colored different levels of grey in the rendering area. If a part is picked, it becomes pink. If a part is in collision the color is red. If you are attached to the part, the part is blue.

You can color the Collidable Parts or Color them only when the mouse goes over them. To color the Collidable Parts go on the Menu Option, select "Workspace Colors" and choose between "Color Collidable Parts" or "Color Collidable Parts on Mouse Over". If you want come back to no color on the Collidable Parts, Select "No Collidable Coloring".