



Pilot Tool for Linking Ship Design to Shipyard Simulation

Principal Investigators:

**Dr. Clay Walden
CAVS Extension
Mississippi State University**

**Dr. Allen Greenwood
Department of Industrial and Systems Engineering
Mississippi State University**

April 20, 2011

Approved for public release

Project Title: Pilot Tool for Linking Ship Design to Shipyard Simulation

Technical Contact:

David Hiscox

Project Engineer
VT Halter Marine
(228) 712-2261, d.hiscox@vthm.com

Principle Investigators:

Dr. Clay Walden

Director, CAVS Extension
Mississippi State University (MSU)
(601) 407-2713, walden@cavse.msstate.edu

Dr. Allen Greenwood

Professor, Industrial and Systems Engineering
Mississippi State University
(662) 325-7216, greenwood@ise.msstate.edu

Funding:

National Shipbuilding Research Program (NSRP)

Period of Performance:

October 2009 – April 2011

This project is performed by Mississippi State University (Center for Advanced Vehicular Systems - Extension and the Department of Industrial & Systems Engineering) for the National Shipbuilding Research Program (NSRP).

TEAM MEMBERS

Mississippi State University

Dr. Clay Walden (Co-PI)
(601) 407-2713
walden@cavse.msstate.edu

Dr. Allen Greenwood (Co-PI)
(662) 325-7216
greenwood@ise.msstate.edu

Travis Hill
(601) 407-2734
thill@cavse.msstate.edu

Robbie Holt
(601) 407-2735
rholt@cavse.msstate.edu

Chase Saunders
(601) 407-2731
csaunders@cavse.msstate.edu

VT Halter Marine

David Hiscox
(228) 712-2261
d.hiscox@vthm.com

ShipConstructor

Patrick Roberts
(251) 340-6200
patrick.roberts@shipconstructor.com

TABLE OF CONTENTS

1	OBJECTIVE AND BACKGROUND	6
1.1	OBJECTIVE	6
1.2	BACKGROUND.....	6
1.3	SOFTWARE.....	8
1.3.1	Flexsim- 3D Modeling Engine.....	8
1.3.2	<i>ShipConstructor</i> Software.....	9
2	PRODUCTION DESIGN SIMULATION TOOL	11
2.1	OVERVIEW	11
2.2	PROBLEM	11
2.3	APPROACH.....	11
2.4	ARCHITECTURE.....	12
2.4.1	Transfer Library	13
2.4.2	Intermediate Database.....	14
2.4.3	Extraction Library.....	14
2.4.4	Administrative GUI	15
2.4.5	Issues/Limitations	16
3	OUTFITTING PROTOTYPE SIMULATION.....	16
3.1	OUTFITTING OVERVIEW	16
3.2	MODEL DESCRIPTION.....	17
3.3	MODELING CONCEPTS	19
3.3.1	Erection	19
3.3.2	Unit/SWBS Outfitting Concept	19
3.3.3	Craft Work Time Concept	22
3.4	MODEL ASSUMPTIONS	24
3.5	POTENTIAL OUTPUTS	25
3.6	ISSUES/LIMITATIONS.....	25
4	FUTURE RESEARCH.....	25
	REFERENCES.....	27
	APPENDIX A: SYSTEM REQUIREMENTS	28
	APPENDIX B: PILOT DESIGN SIMULATION TOOL USAGE	29
	APPENDIX C: INTERMEDIATE DATABASE XML SCHEMA	30

TABLE OF FIGURES

FIGURE 1: NSRP PANEL LINE SIMULATION 7
 FIGURE 2: DECISION SUPPORT SYSTEM..... 8
 FIGURE 3: SHIPCONSTRUCTOR PROJECT EXAMPLE..... 9
 FIGURE 4: SHIPCONSTRUCTOR HVAC SYSTEM..... 10
 FIGURE 5: SYSTEM SEQUENCE DIAGRAM..... 12
 FIGURE 6. LINKAGE TOOL ARCHITECTURE 13
 FIGURE 7: ADMINISTRATIVE GUI 15
 FIGURE 8: GROUP 6 SWBS OUTFIT AND FURNISHINGS 16
 FIGURE 9: VT HALTER SHIPYARD LAYOUT 17
 FIGURE 10: NRSP OUTFITTING PROTOTYPE MODEL 17
 FIGURE 11: PLATEN AREA AND CRAWLER DELIVERY..... 18
 FIGURE 12: OUTFITTING PROTOTYPE FLOW DIAGRAM 18
 FIGURE 13: NSRP OUTFITTING PROTOTYPE ERECTION PROCESS 19
 FIGURE 14: FLEXSIM GT_SWBS TABLE 20
 FIGURE 15: CUSTOMER SPECIFIC DATABASE PART A 20
 FIGURE 16: CUSTOMER SPECIFIC DATABASE PART B 21
 FIGURE 17: USER COMMAND: "CALCWORKTIME" 21
 FIGURE 18: SWBS TO CRAFT RELATIONSHIP 22
 FIGURE 19: USER COMMAND: "GETCRAFTWORKPERCENTAGE" 23
 FIGURE 20: SWBS TO UNIQUE CRAFT DATABASE TABLE 23

TABLE OF TABLES

TABLE 1: LINKAGEDATA.TXT FILE LEVELS 14
 TABLE 2: INTERMEDIATE DATABASE CLASSES 14

1 OBJECTIVE AND BACKGROUND

The following sections provide the overall project objective and the background required to understand the production design simulation tool and the outfitting prototype. This work was conducted in close cooperation with V.T. Halter Shipyard, which served as the prime shipyard partner throughout the project.

1.1 Objective

The objective of this project is to develop a Pilot Design Simulation Tool (PDS Tool) and develop a pilot outfitting prototype simulation model.

The PDS Tool is intended to enhance existing engineering software tools and enable the linkage of key elements of a ship's design to a simulation model. It is proposed that this tool will enhance the value of and extend the life of future and existing simulation models by incorporating up-to-date design data into the simulation model. In addition, this tool targets the elimination of a substantial data entry step that is currently required to run simulation models of shipyards. This is particularly a challenge for the shipbuilding industry due deep and complex product structure. Therefore this project focuses on developing a tool that which targets minimizing this problem in the DSS and other simulation model and analysis projects. For example, the design data for a new mid-size ship (e.g., barge) could easily require one week or more of data entry time in order to fully populate data required to support a fabrication shop simulation model. Of course, frequently yards deal with more than one ship hull in process at any one time, which makes the data requirement even more daunting. Also, the data entry problem is increased when developing simulation models of outfitting activities, which are highly dependent on design custom requirements. Therefore, there is a significant need to develop a Pilot Design Simulation Tool (PDS Tool).

While numerous simulation models have been developed over the last several years, most have focused on developing models of fabrication. Very few examples are available regarding how to best model the highly variable processes supporting outfitting. Since outfitting is represents a major element of the shipbuilding enterprise, it is very important to the industry to develop an approach for modeling this labor intensive processes. Of course, in order to develop a shipyard wide simulation model both fabrication, erection and outfitting are required. This project proposes to evaluate and develop approaches to address the outfitting modeling challenge.

1.2 Background

The creation of a simulation model gives a decision maker the ability to view a simplified version of a real world system. Modifications can be made quickly and inexpensively to the simulation model in order to gain valuable knowledge about the affects the changes will have on the overall system. The decision maker can make various changes and use the results to gain a better understanding of how the system will react to certain modifications. Once the best design for the simulated system has been determined, the changes can be made in the real world system.

As part of a previous project [1], a Shipyard Planning Decision Support System (SPDSS) was developed to rapidly estimate the impact of proposed ships given work already in progress. It was used to perform constraint-based analysis which aids in the identification of bottleneck operations and excess capacity, and helps estimate ship completion dates. All of these help planners find and exploit opportunities during early stages of the planning process.

The system was design to allow planners to utilize the power of simulation modeling analysis without being experts in the computer and simulation technologies. At the core of the system resides a simulation model of combine shop operations (material prep, panel line, and unit assembly). The simulation model incorporates all product routings, resources, shifts and downtimes to accurately represent the real system. The screenshots of the SPDSS and simulation model can be seen in Figure 1 and the basic architecture of the decision support system can be viewed in Figure 2.

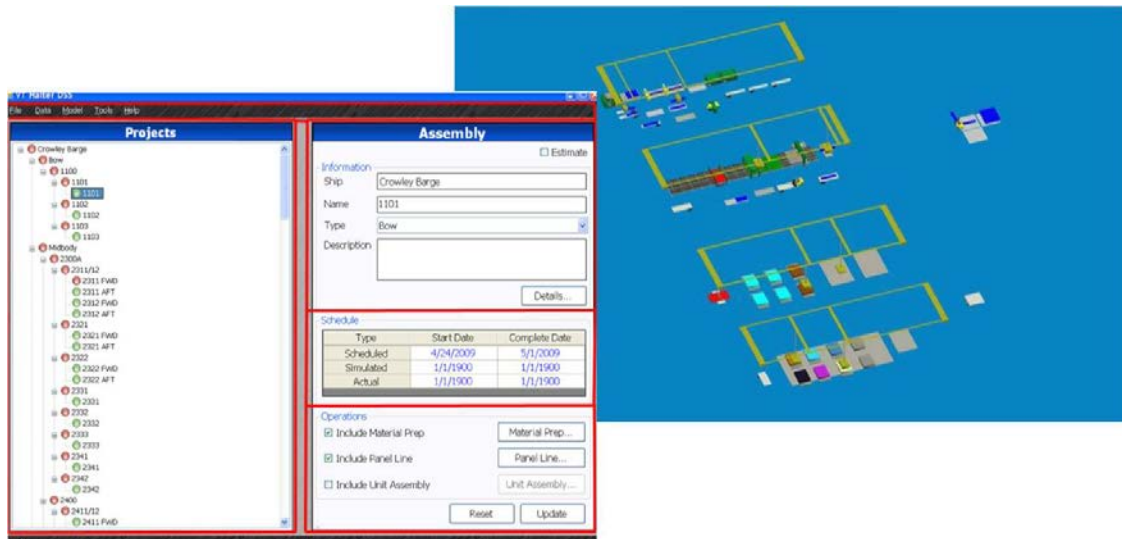


Figure 1: NSRP Panel Line Simulation

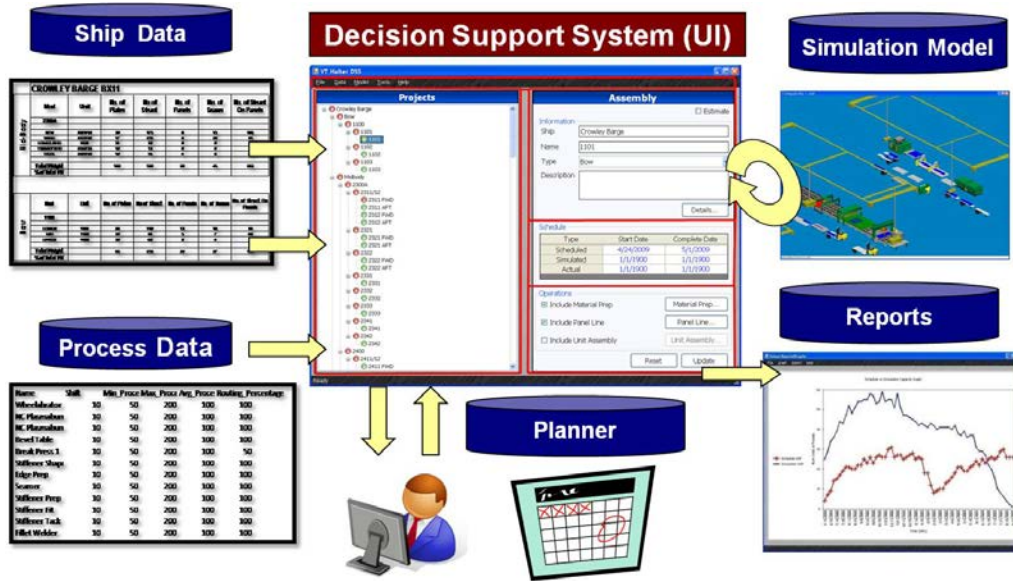


Figure 2: Decision Support System

Although the system accomplished everything it set out to, during the development, it became evident that there was a high level of maintenance required in terms of manual data manipulation and entry to keep the system feed with up-to-date information.

1.3 Software

1.3.1 Flexsim- 3D Modeling Engine

Flexsim is a discrete event simulation software package that is used for modeling, analyzing, visualizing and optimizing processes. The software is extremely robust and can be manipulated to model virtually any process. *Flexsim* makes it easy to pinpoint bottlenecks of current systems as well as analyzing proposed system design changes. A simulation can run in real-time or at an accelerated time. This means that a simulation run may represent a year, but only take a few minutes to actually run on a computer. This provides the ability to make changes to a virtual system and view the effects quickly and less costly than actually making changes to the real system.

Although *Flexsim* has many predefined objects and options, there is a need to develop shipyard industry specific constructs that ultimately reduce the model development time. *Flexsim* does provide its own scripting language that is object-oriented based and similar to the C++ language which enables these industry specific models to be developed. *Flexsim* also allows you to read data in directly from Microsoft Access databases and Excel files. Layouts created in AutoCAD can be imported into a simulation model in order to create a scaled environment of the real world system. This is important because travel times for task executers (ex. Operators, crawlers, and cranes) are dependent on the distance traveled. This helps to increase the accuracy of the utilization calculated for each of the task executers.

1.3.2 *ShipConstructor* Software

ShipConstructor is a CAD/CAM software specifically built to address the requirements of the shipbuilding industry. ShipConstructor utilizes AutoCAD as a visualization and design interface while retaining all of the data pertaining to ship structures in a Microsoft SQL database. Beyond the basic CAD geometry required for describing the ship systems, there is also all of the production sequencing information relevant for vessel construction.

The ShipConstructor suite of software includes all of the major ship systems and allows designers to work concurrently in a single production model. The various available modules are:

- Hull
- Structure
- Pipe
- HVAC
- Equipment

With the ability to also include cross-disciplinary information as well, such as pipe connections available on equipment components and pipe penetration marking and tracking in structural elements.

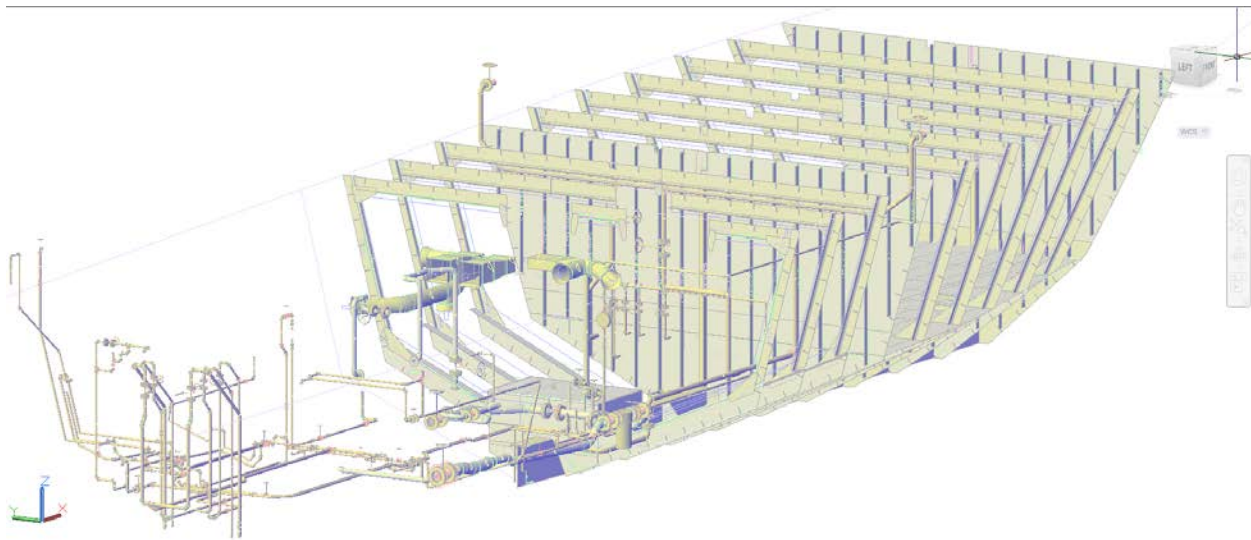


Figure 3: ShipConstructor project example

The ShipConstructor Hull module allows for the design and creation of curved plate and twisted stiffeners. The Structure module allows for detailed structural design to be created utilizing a library of standard structural shapes and parts as defined by the shipyard, and for the production build sequencing to be specified.

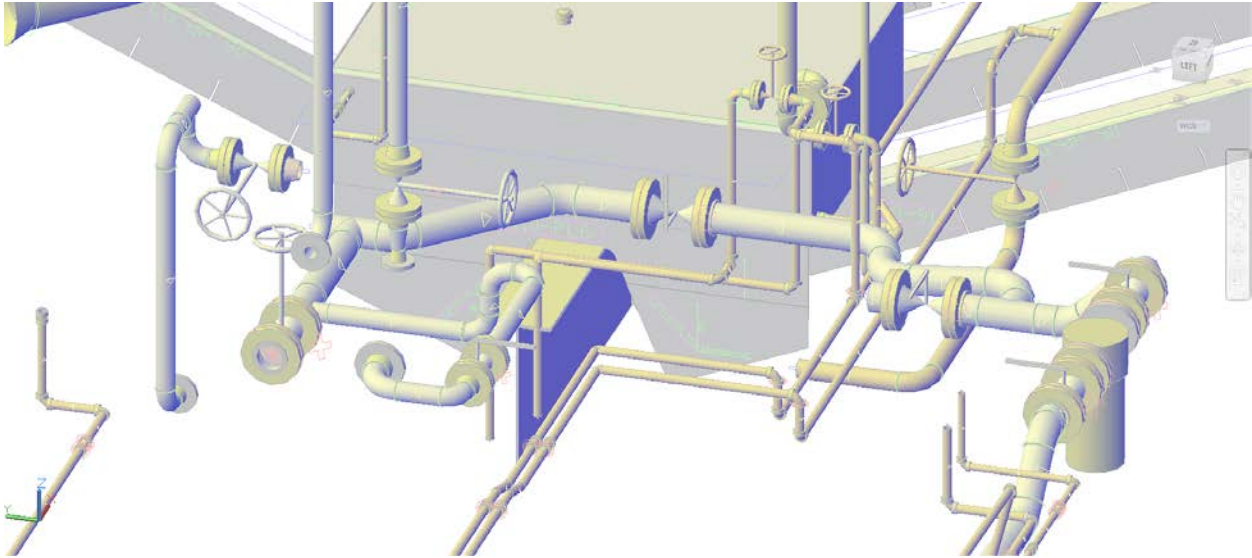


Figure 4: ShipConstructor Pipe System

The Pipe and HVAC modules allow for complex, spec-driven distributed systems to be routed through the structural model of the ship in-place, and to include the spool breakdowns for these systems to be included in the appropriate build sequencing level for ship production.

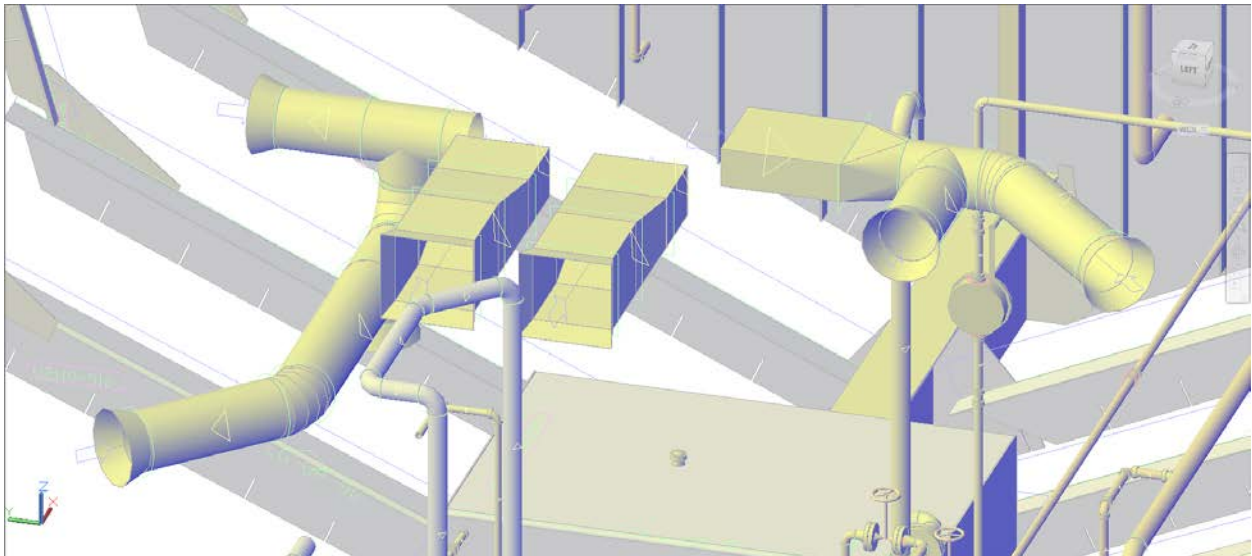


Figure 5: ShipConstructor HVAC System

The Product Hierarchy in ShipConstructor allows for the complete description of all the required build sequencing that has to occur to construct the vessel. From the full ship through each unit, assembly, sub-assembly and block all the way down to the individual piece parts. Every stage of vessel construction can be specified. The complete ship model as described in a ShipConstructor project fully defines all of the structural and distributed systems as they should exist in the final

delivery stages of a vessel, and more importantly all of the manufacturing sequencing stages as well.

ShipConstructor also has an Application Programming Interface (API) that allows third-party tools to query all of the underlying data behind a ShipConstructor project. This includes all of the geometric data as well as the product hierarchy data.

2 PRODUCTION DESIGN SIMULATION TOOL

2.1 Overview

The Production Design Simulation Tool (PDS Tool) is a pilot software tool intended to bridge a gap between ship design and production engineering. To accomplish this data from ship CAD needs to be made available to other engineering tools. This project focuses on linking CAD data from *ShipConstructor* to the discrete event simulation package *Flexsim* to aid the decision making process through simulation analysis. Although this project utilizes *ShipConstructor* as the CAD software and *Flexsim* as the simulation software, the concept and core software developed here are not limited to these packages.

2.2 Problem

Simulation modeling and analysis has been proven to be a key engineering tool to evaluate and improve production systems in various domains including shipbuilding. Unfortunately, one of the biggest issues with simulation modeling can be the getting data to feed the model [2]. In some instances this is accomplished through a great deal of manual data manipulation which is required because different systems such as CAD and discrete-event simulation package not being able to communicate with each other. This project targets this obstacle using a proto-type software tool linking *ShipConstructor*, as the CAD system, and *Flexsim*, as the simulation system.

2.3 Approach

The PDS Tool is designed to transfer data from *ShipConstructor* to *Flexsim* while being modular enough to accept future libraries for other simulation packages and CAD packages. The appropriate library just needs to be developed for a specific simulation package or CAD package. The requirements have been defined in the Linkage Tool Requirements document [3].

Essentially the PDS Tool is set up as pull system as shown in Figure 6. The simulation model initiates a call to pull data for the simulation model. The “age” of the data in the Intermediate Database is evaluated to determine whether the data is recent enough to use or needs to be refreshed. If the data is recent, the Transfer Library begins transferring data from the Intermediate Database into the simulation model. If the data needs to be refreshed, the Intermediate Database calls the Extraction Library to update the database from *ShipConstructor*.

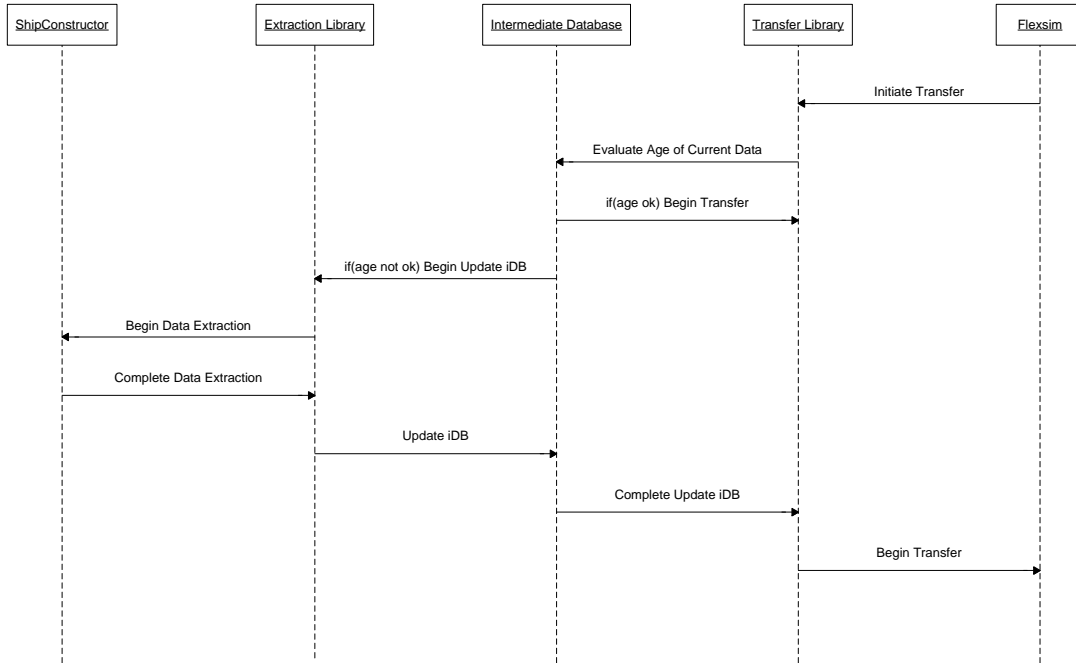


Figure 6: System Sequence Diagram

2.4 Architecture

The PDS Tool is comprised of 4 key components: the Transfer Library, the Intermediate Database, the Extraction Library, and the Administrative Graphical User Interface (GUI). These components are developed in a mixture of C++ and C#. This section will cover each of these components and their relationship to each other. The basic architecture can be seen in Figure 7.

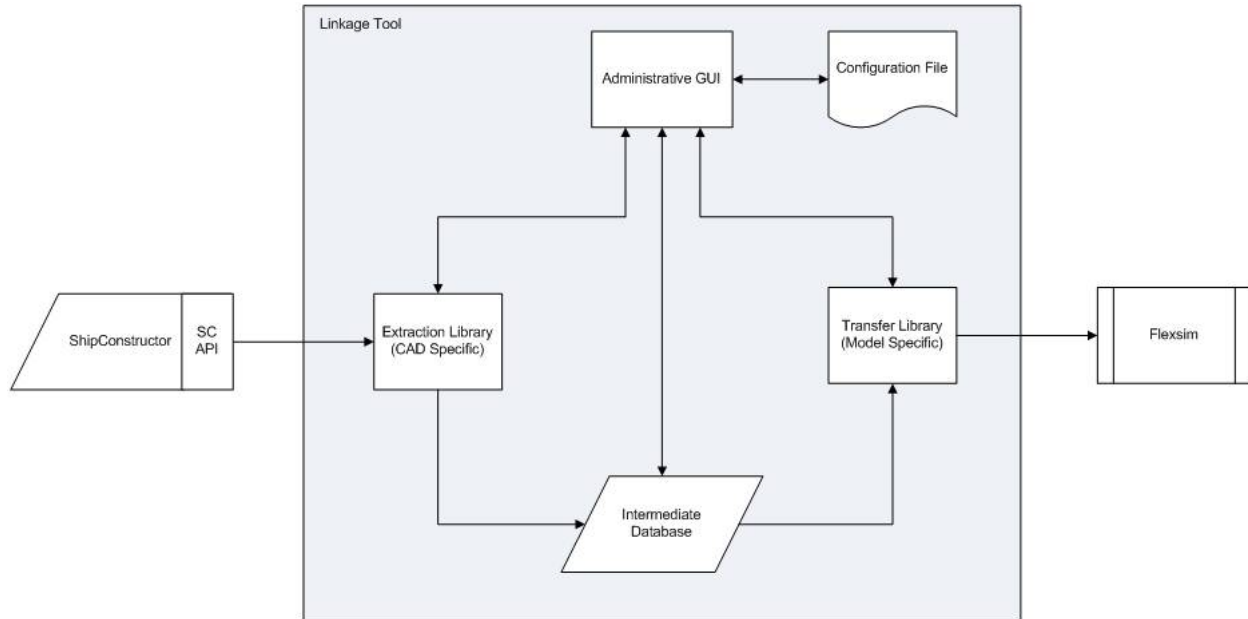


Figure 7: Linkage Tool Architecture

2.4.1 Transfer Library

The Transfer Library (`dataTransfer.dll`) is responsible for pulling data from the Intermediate Database and transferring it into the simulation model. This library is developed in native C++ to accommodate *Flexsim* which is also developed in native C++. The library is used by the simulation model in conjunction with an external text file (`LinkageData.txt` – discussed later) to pull the information, at the appropriate level of detail, from the Intermediate Database. The Transfer Library parses the Intermediate Database and loads the data into a global table (`SC_Data`) in *Flexsim*. The simulation model then uses this global table to create the appropriate objects for the model.

The required level of detail and the amount of data to be transferred depends on the system being modeled (i.e. if unit level data is all that is required to perform a given analysis there is no point in transferring in detailed subassembly information when a roll up of that data will suffice). Because of this, the `LinkageData.txt` file is provided to define the appropriate level of detail. This file contains a single line of text. This line tells the Transfer Library to add data at a specific level (e.g. unit). This file is generated by the user for each model that uses the linkage tool and is stored in the same directory as the model. The valid values in this file are shown in Table 1 and are case-sensitive.

Table 1: LinkageData.txt File Levels

Data Levels	Description
ship	Ship specific data and roll up information such as number of units.
ship/unit	Unit specific data such as number of assemblies and roll up information about number of plates, pipes, etc. as well as ship unit parent information.
ship/unit/assembly	Assembly specific data such as number of plates, pipes, etc. as well as ship and unit parent information.
ship/unit/assembly/sub	Used primarily for structure data. This level includes subassembly specific data including plate and extrusion data as well as ship, unit, assembly parent information.
ship/unit/assembly/swbs	Used primarily for outfitting data. This level includes SWBS specific data including pipe and HVAC data as well as ship, unit, assembly parent information.

2.4.2 Intermediate Database

The Intermediate Database is a library component and eXtensible Markup Language (XML) file. The library provides a common set of classes and functions to be used by the Transfer Library, the Extraction Library, and Administrative GUI. It provides the following key classes shown in Table 2.

Table 2: Intermediate Database Classes

Data Levels	Description
Ship	Container object for ship information. Contains sub-objects: Unit, Assembly, SubAssembly, SWBS, Plate, Extrusion, CurvedPlate, CorrugatedPart, TwistedExtrusion, Pipe, PipePart, HVAC, HVACPart, Equipment.
iDB	Provides methods for updating the Intermediate Database and settings.
iDBWriter	Provides methods to write to the iDB.xml file.
SC_DataSettings	Provides setting information for ShipConstructor such as server name, username, password, and a list of projects to load.

The XML file (iDB.xml) is the repository of the data extracted from the CAD software. It is based on the XML Schema shown in Appendix C and resides in the PDS Tool directory. The main purpose for the XML file is to mitigate the need to constantly access the massive *ShipConstructor* database for information that may not have been updated recently or relevant to the simulation model. In addition, it provides an opportunity for third-party applications to add additional data such schedule or process lane information.

2.4.3 Extraction Library

The Extraction Library is responsible for communicating with and extracting most the data defined in the Fabrication and Outfitting Requirement documents [5,6]. *ShipConstructor* provides an Application Programming Interface (API) that gives access to the core

ShipConstructor database. This library utilizes the API to retrieve the relevant data and send it to the Intermediate Database for storage.

2.4.4 Administrative GUI

The Administrative GUI allows a user to set up the *ShipConstructor* database settings such as server name, username, password, and project list. Figure 8 shows the main interface. The interface is divided into three tabs. The “Configuration” tab allows users to specify *ShipConstructor* database information. The “Projects” tab allows the user to select which projects to extract data from. This list is automatically filled based on the server selected on the “Configuration” tab. The “Data” tab provides methods to perform the data extraction process and view the current iDB.xml file. The data extraction process can require hours of processing depending the number and size of the projects being loaded into the Intermediate Database.

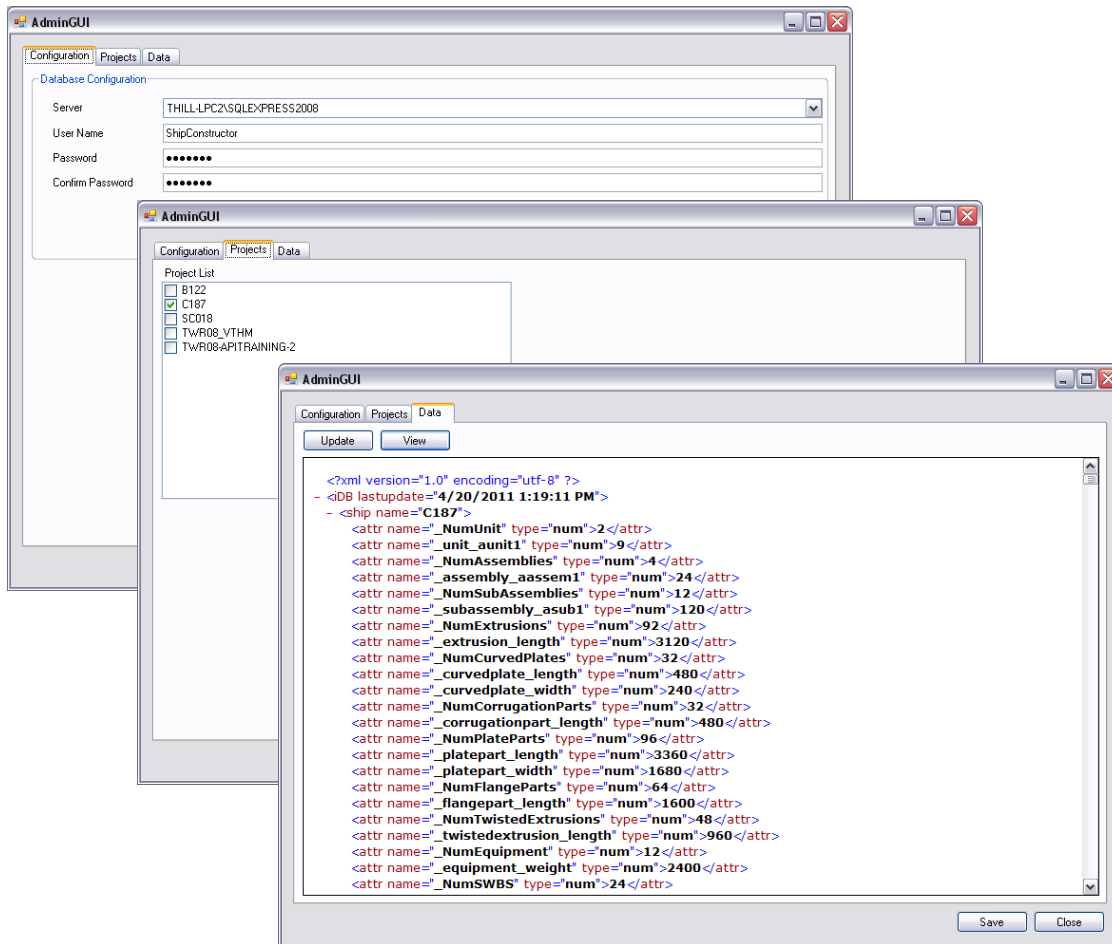


Figure 8: Administrative GUI

2.4.5 Issues/Limitations

The following items limited the structure of the PDS Tool:

1. No implementation and testing could be performed with the electrical module of *ShipConstructor* 2009, due to the absence of electrical data in files obtained from shipyard.
2. HVAC has limited implementation in this pilot tool because of limited data availability due to current business practices at the partnering shipyard (e.g., HVAC design and installation).
3. Only allows ship project information to be on a single instance of Microsoft SQL Server.

3 OUTFITTING PROTOTYPE SIMULATION

3.1 Outfitting Overview

Outfitting is the process of installing systems into a fabricated Unit. A Ship Work Breakdown Structure (SWBS) is a 3-digit identifier used to uniquely identify each of these systems [4]. These systems include prefabricated pipe details and plumbing fixtures, HVAC ducts and equipment, electrical wires and systems, structural elements (e.g. ladders, rails, walkways, foundations, etc.) and the painting of surfaces and components (internal and external). A short list of SWBSs and the systems that are included in the respective SWBS can be found in Figure 9. The SWBS number 611, for example references the Hull fittings on the ship, so each unit that requires hull fittings will have this SWBS associated with it.

Group 6 Outfit and Furnishings	
600	Outfit and Furnishings, General
601	General Arrangement-Outfit and Furn.
602	Hull Designating and Marking
603	Draft Marks
604	Locks, Keys, and Tags
605	Rodent and Vermin Proofing
610	Ship Fittings
611	Hull Fittings
612	Rails, Stanchions, and Lifelines
613	Rigging and Canvas

Figure 9: Group 6 SWBS Outfit and Furnishings

The process of outfitting is performed in the following areas: the combine shop, the platen area, the shipway, and in the water after departing the shipway.

3.2 Model Description

This is a proto-type simulation model for depicting outfitting processes performed in the following process lanes: Platen Area and Shipway Area as show in Figure 10. The start of this outfitting prototype model, shown in Figure 11 is the end of the fabrication shop. The shop floor is not modeled; however, it is taken into account that some percentage of the outfitting process was completed in the fabrication shop. That portion is represented by a percentage complete of the total SWBS system prior to entering the platen area.



Figure 10: VT Halter Shipyard Layout

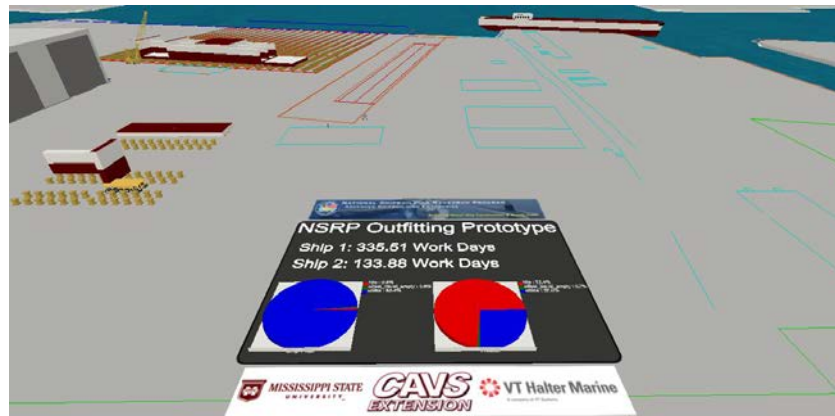


Figure 11: NRSP Outfitting Prototype Model

A unit enters the system based on an erection sequence found in the customer specific database. This database contains the following: an erection sequence, list of crafts specific to the customer's shipyard, identification of crafts associated with each SWBS, estimated work times, and description of work by craft, on a specific SWBS in each process lane (shop, platen, shipway, etc.). . A crawler transfers the unit from the shop and delivers it to the platen area. At the platen area, depending on the SWBS installed in the unit, craft operators perform work on the

unit. Once all work required for this process lane has been completed on the unit, a crawler transfers it to a location where it meets a crane as show in Figure 12.

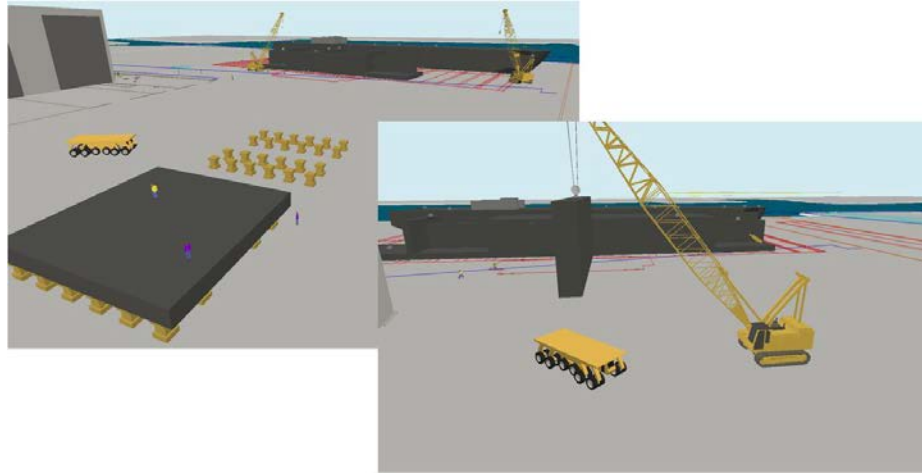


Figure 12: Platen Area and Crawler Delivery

The crane picks up the unit from the crawler and takes it to the shipway. The crane places the unit into position where it is installed. Once the erection process for the unit has been completed, the crane becomes available for the next unit to be erected on the shipway. Work that needs to be completed, based on the SWBS being installed is performed. Once all units have been installed and all outfitting work completed, the ship is moved to the water and exits the system. The diagram of the flow of units through the system is found Figure 13.

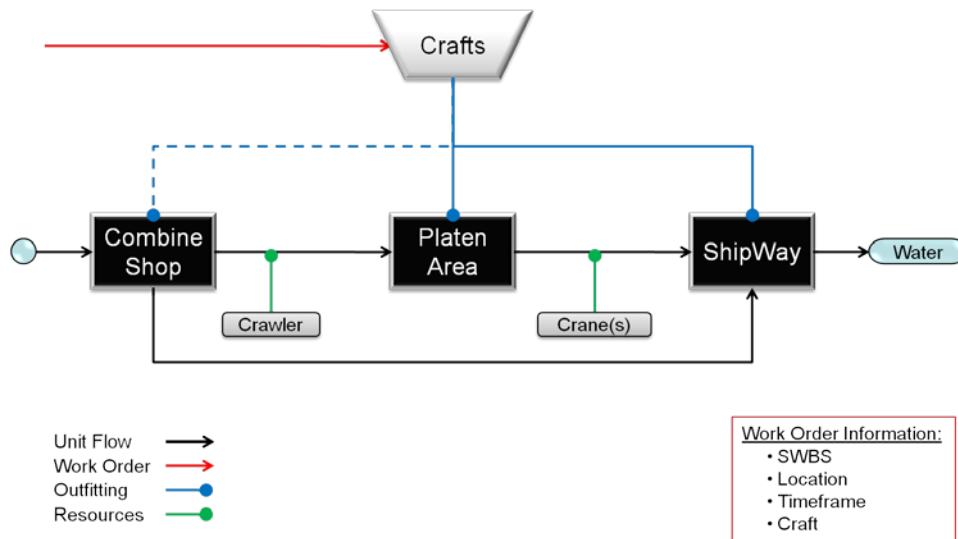


Figure 13: Outfitting Prototype Flow Diagram

3.3 Modeling Concepts

3.3.1 Erection

Once all work has been completed as described in the SWBS, the crawler transfers the unit to a transfer location. A crane arrives and picks up the unit at that location, the crane then proceeds with the unit to a shipway and places the unit into the correct position based on position values located in the “positioning table.” This global table contains information about each unit that makes up the completed ship. While the crane is holding it in place, the unit is joined to other already installed units. It then becomes available for outfitting on the shipway. In Figure 14, a unit can be seen being installed on the first shipway location.

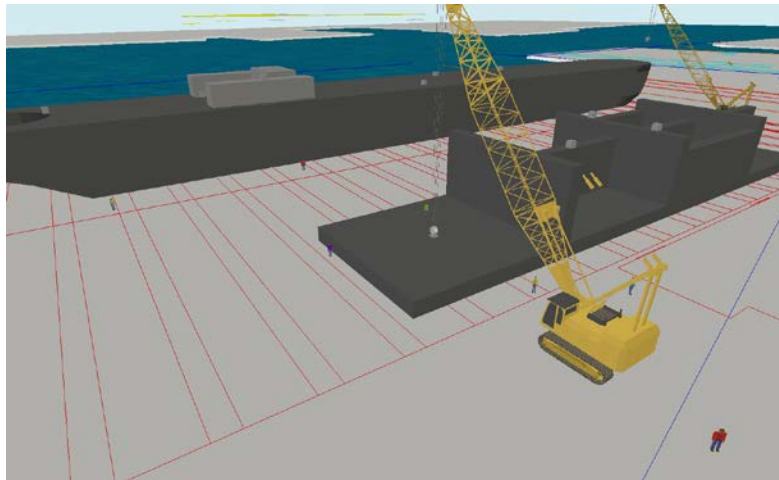


Figure 14: NSRP Outfitting Prototype Erection Process

There are two shipway locations in this prototype model where the erection and outfitting processes are performed. These two locations are identical, which allows for multiple ships to be built (i.e. erected and outfitted) simultaneously. The model contains references a “global table”, which identifies which shipway a given ship is being built at anytime during the simulation. This table is used by the crane to determine where to deliver and install a particular unit.

3.3.2 Unit/SWBS Outfitting Concept

When a unit enters the system, the model retrieves data on all of the SWBSs that need to be installed in this unit. Once it has the SWBS information needed, the total work time to install a SWBS in a particular unit is calculated using work time estimation values found in the Customer Specific Database. This database has values for each process step associated with each SWBS. Each step has a minimum and maximum estimation factor value, based on type of ship (e.g. barge, military vessel, etc.). A uniform distribution is applied to these two values in order to determine a factor value for the process step. Each step also contains a value(s) that represents the type of information needed about the unit to calculate a work time (e.g. linear ft. of pipe is needed to calculate time to install pipe in unit).

Once the type of information needed has been determined, the actual value(s) is retrieved from the “global table” that contains information about each unit. This value(s) is multiplied by the value to determine the work time for the specific process step of the SWBS. A SWBS can be made up of multiple processes, so once a work time has been generated for each process, the work times are summed to determine the total work time for the SWBS. The user provides standard percentages of outfitting to be completed at each process lane. This information is used to divide the total work time among the individual process lanes.

3.3.2.1 Example of Unit/SWBS

An example of this process is outlined below:

Unit number 2312 enters the system based on the erection sequence. The global table, “gt_SWBS”, is referenced to determine exactly which SWBS are installed in the unit. The SWBSs that need to be installed are 202, 235, 244, and 320, which can be seen in Figure 15.

gt_SWBS										
		Rows: 614.000		Columns: 125.000		Clear on Reset				
Unit #	SWBS	Circumferenc	diamater	linear ft.	Horsepower	Kilowatts	lbs	Letters	Lin. F	
2312.000	202.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
2312.000	235.000	0.000	0.000	5.000	0.000	0.000	0.000	0.000	0.000	
2312.000	244.000	5.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
2312.000	320.000	0.000	2.000	40.000	0.000	0.000	0.000	0.000	0.000	

Figure 15: Flexsim gt_SWBS Table

SWBS #320 is composed of 3 different lines in the customer specific database, which can be seen in Figure 16. Each row has its own set of factor values and at the end of each row there are columns which contain reference values. These values reference the column location of the information values (e.g. circumference, diameter, linear ft., etc.) in the global table “gt_SWBS.”

SWBS_to_WorkTime_Filtered										
ID	SWBS	Description	Commerical	max1	Commercial	max2	Commercial	max3	M	
12	320	EXHAUST -MAII	1.8	2.2	1.8	2.2	1.8	2.2	2.2	
13	320	EXHAUST -MAII	0.8	1.2	0.8	1.2	0.8	1.2	1.2	
14	320	EXHAUST -MAII	0.2	0.3	0.2	0.3	0.2	0.3	0.3	
15	321	EXHAUST -GEN	1.8	2.2	1.8	2.2	1.8	2.2	2.2	

Figure 16: Customer Specific Database Part A

The factor value used is dependent on the type of ship being produced. In this case, the type is a commercial barge, so for the first row corresponding to SWBS #320 in the Customer Specific Database, the minimum and maximum factor values are 1.8 and 2.2. A uniform distribution is applied using these values as the lower and upper boundary to determine the factor value to be used in the model. Using the reference information, seen in Figure 17, the global table,

“gt_SWBS”, is accessed from the simulation model in order to determine the values that need to be multiplied by this factor value.

Units Info	Notes	Col 1	Col 2
2.2 *(Diameter) = I	Silencer	4	65
1.2 *(Diameter) = I	Flex. Conn	4	66
0.3 *(Diameter of I	Pipe	4	5
2.2 *(Diameter) = I	Silencer	4	65

Figure 17: Customer Specific Database Part B

The reference values are 4 and 65, so in “gt_SWBS” on the row for unit 3212 and SWBS #320 the values found in column 4 (circumference) and 65(# of silencers) are obtained and multiplied by the factor value to get a time value (units are in minutes). The code that is executed to perform this can be found in Figure 18.

```

13 //Pull Information from the SWBS_to_Worktime database given the SWBS information and put it in a temporary
14 string cmd = concat("select * from [SWBS_to_Worktime_Filtered] where SWBS = ", numtostring(SWBS,0,0));
15 dbsqlquery(cmd);
16
17 //Look up in temporary table and get which variables to pull to use for calculation
18 for(int k = 1; k <= dbgetnumrows(); k++)
19 {
20     for(int j = 16; j <= dbgetnumcols(); j++)
21     {
22         if(dbgettablenum(k,j) > 0)
23         {
24             if(factor == 0)
25             {
26                 factor = gettablenum("gt_SWBS",tblrow,dbgettablenum(k,j));
27             }
28             else
29             {
30                 factor = factor * gettablenum("gt_SWBS",tblrow,dbgettablenum(k,j));
31             }
32         }
33     }
34     mintime = (dbgettablenum(k, (BOATYPE+2)+2)) * 60; //Converting Hours into minutes
35     maxtime = (dbgettablenum(k, (BOATYPE+2)+3)) * 60; //Converting Hours into minutes
36     pt("Factor: "); pf(factor); pt(" Mintime: "); pf(mintime); pt(" Maxtime: "); pf(maxtime); pr();
37     if(workTime == 0)
38     {
39         workTime = uniform(mintime,maxtime) * factor;
40     }
41     else if(workTime > 0)
42     {
43         workTime = workTime + (uniform(mintime,maxtime) * factor);
44     }
45     factor = 0;
46 }
47
48 pt("Work Time: "); pd(workTime); pr();
49
50 getcraftcodes(SWBS, workTime, area, ID, sendingobject);

```

Figure 18: User Command: "calcworktime"

There are 3 rows in the Customer Specific Database for SWBS #320, so the process for calculating a time is repeated for each row using the same method described above. Once all

time values have been calculated, they are summed up to get a total amount of time to install the SWBS in unit 3212. This value must be divided between the Shop, Platen Area, and Shipway. The customer specific database provides the percentages used to calculate the division of work done at each area.

3.3.3 Craft Work Time Concept

Once the work times have been determined, information containing which crafts are needed to install the SWBS in the given unit in a given process lane is gathered from the customer specific database. This is the final piece of information needed to complete the work order, which contains the following information: unit number, SWBS being installed, process lane, and work time to install SWBS. The work order is released to the craft specific foreman (dispatcher), which determines how to dispatch its operators to perform the work listed. There is a separate work order for each SWBS per craft per area per unit. In Figure 19, it shows the information that is obtained from the relationship of SWBSs to the craft code table. The craft name column provides the craft type (shipfit, weld, etc.) and location (shipway, platen, etc.) where the work is performed. In this project, only the crafts associated with the platen and shipway are considered.

SWBS_to_Craft		Craft Code	
SWBS	Craft	CRAFT #	CRAFT NAME
610	153	152	SHIPFIT-PANEL LINE
610	164	153	SHIPFIT-FAB SHOP
610	175	164	SHIPFIT-PLATEN
610	176	175	SHIPFIT-SHIPWAY
610	203	176	SHIPFIT-WETDOCK
610	204	180	OPERATOR-MATERIAL HANDLING
610	215	202	WELD-SERIES ARC
610	216	203	WELD-FAB SHOP
610	243	204	WELD-PLATEN
610	306	215	WELD-SHIPWAY
		216	WELD-WETDOCK

Figure 19: SWBS to Craft Relationship

The total work time for the SWBS is divided among the associated crafts randomly. The number of operators that can be sent to work on the unit is limited due to space constraints within the unit. Once the amount of work time required has been formulated, the “dispatcher” assigns a time to perform work on the unit. This time should be based on the budget hours associated with each task within a particular SWBS. However, this data was not received so that a “place-holder” routine was used within the model which can be updated once that data is available from the partnering shipyard. . The overall work time is divided by the number of people assigned to the work (increasing people, reduces the time to complete the work order).

3.3.3.1 Example of Craft Work Time

The example described in the previous section continues below:

Once the total work time required to install SWBS #320 in unit 3212 has been calculated, it must be distributed among the crafts associated with the SWBS. The user command “getcraftworkpercentage”, seen in Figure 20, is used to distribute the total work time among the unique craft numbers needed to install the SWBS.

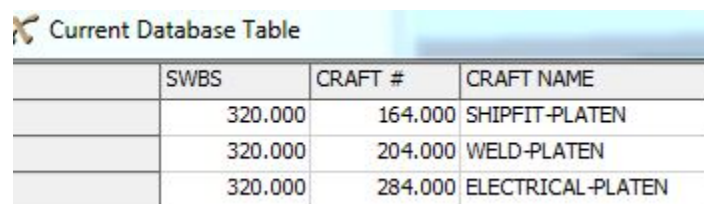
```

14 sqlcommand(area, SWBS);
15
16 areaworkpercentage = getareaworkpercentage(ID, SWBS, area);
17 workTime_area = workTime * areaworkpercentage;
18 pt("Work Percentage: ");pf(areaworkpercentage);pr();
19 pt("Work Time for ");pt(area);pt(": ");pf(workTime_area);pr();
20
21 for(int j = 1; j <= dbgetnumrows(); j++)
22 {
23     craft = dbgettablenum(j,2);
24     for(int k = 1;k <= gettablerows("gt_craftcode_port");k++)
25     {
26         if(gettablenum("gt_craftcode_port",k,1) == craft)
27         {
28             portnum = gettablenum("gt_craftcode_port",k,2);
29         }
30     }
31     craftPerc = getcraftworkpercentage(ID, SWBS, craft);
32     craftWorkTime = workTime_area * craftPerc;
33     inc(label(sendingobject,numtostring(craft)),1);
34     senddelayedmessage(centerobject(sendingobject, portnum), .1, sendingobject, craftWorkTime, ID, craft);
35     pt("Craft: ");pd(craft);pt(" Work Time per Craft: ");pf(craftWorkTime);pr();
36 }
-- ..

```

Figure 20: User Command: "getcraftworkpercentage"

This user command references a table, seen in Figure 21 is generated based on the information in the database. This table lists the crafts associated with the specific SWBS of interest. In this case, SWBS #320 needs the following craft numbers: 164, 204, and 284.



	SWBS	CRAFT #	CRAFT NAME
	320.000	164.000	SHIPFIT-PLATEN
	320.000	204.000	WELD-PLATEN
	320.000	284.000	ELECTRICAL-PLATEN

Figure 21: SWBS to Unique Craft Database Table

The work time information for each craft is sent to their respective “dispatcher.” The “dispatcher” assigns a the number of its available operators to perform the work. This “assignment” needs to be updated once the SWBS budget data is available.

3.4 Model Assumptions

The following are major assumptions incorporated in the model:

1. Speed of crane is equivalent to speed of crawler, which is 107.29 meters/minute (4 mph).
2. Crane lift speed = 2 m/min.
3. Crawler acceleration and deceleration = 2 m/min.
4. Load and unload time on crawler = 5 min.
5. Time to secure unit to crane = 30 min (the crane loads the unit delivered from the crawler).
6. Unload time of crane is dependent on the specific unit being positioned.
7. Craft workers walk at a rate of 80.47 m/min (3 mph).
8. Operators in the model represent a unique craft code.
9. An operator in the model represents a group of a specific craft and the total number in the group can vary depending on a label set on the operator.
10. There are 4 platen areas in the model.
11. Every unit goes to the platen area before going to the shipway.
12. The model is fed an erection sequence that dictates the order of flow throughout the model.
13. There is space for 2 ships to be built on the shipway simultaneously.
14. Outfitting can be done on the ship when it is in either position 1 or 2 on the shipway.
15. Once all SWBS Systems have been installed in a ship, the ship is launched into the water and exits the system.
16. The customer specific database is in Excel form, which is fed into the access database.
17. The erection sequence that feeds the model was obtained from previous projects worked on.
18. The affect of adding more craft operators to work on a SWBS has a linear effect on the reduction of work time (doubling the resources, reduces the work time by half).
19. A person occupies 9 sq ft of area when working in a unit.

3.5 Potential Outputs

The power of simulation comes from the variety of outputs that can be generated. Modifications can be made to the information that feeds the model, so that outputs from each run can be compared to help develop a better understanding of the entire system. The following are potential outputs that this Outfitting Prototype Simulation Model can provide:

1. Makespan – The time required to complete all units of a given ship from the end of the shop floor down to the water.
2. Craft Utilization – Utilizations of each craft type (e.g. electrical, pipe fitters, etc.).
3. Location Utilization – Utilizations of key locations (e.g. platen, shipway, etc.).
4. Manning Levels – Level of staff required to produce desired results.
5. Craft Bottlenecks – A craft(s) may be a key contributor to increasing the overall makespan.
6. Multiple Ship Interactions/Impacts – Each ship's components compete for the same resources.

3.6 Issues/Limitations

The following items are limitations to the Outfitting Simulation Prototype Model:

1. The prototype model uses a randomly generated number to determine how much work time of a SWBS needs to be performed by a specific craft type. This information should be provided by the customer and include a percent value for each craft required for a given SWBS.
2. A key issue with the model is the way a craft dispatcher decides to utilize its pool of operators. The customer specific database needs to contain ship specific information about the time budget. This critical information can be transformed into logic which will help in the decision making process that the dispatcher goes through in order to determine how many operators to deploy to perform work on a SWBS.

4 FUTURE RESEARCH

The project proved to be successful, but during development, areas of improvement were identified. The following are items that should be investigated in future research:

1. Incorporate the PDS Tool and Outfitting Prototype Simulation Model into a decision support system, which can be used to analyze ship production across the entire shipyard.

2. Expand the PDS Tool and modeling efforts to incorporate a larger variety of ship types. Currently, the model uses information from a single ship.
3. Extend the PDS Tool to utilize additional simulation and CAD packages.
4. Expand the role of the LinkageData.txt file to provide the user with the ability to pull exactly the data need for his/her application. This could reduce data waste within the simulation and ultimately improves model run speed.
5. Expand the PDS Tool to handle multiple instances of Microsoft SQL Server. It is possible for a company to have ship project files across multiple servers.
6. Optimize code to improve performance.
7. Information about craft percent values for installation of a SWBS in a specific unit on a process lane should be obtained and incorporated into the customer specific database. At the time of the project, this information was not obtainable, so depending on which crafts are involved in the work on a SWBS, the total work time was randomly divided among the crafts.
8. Information about craft type time budget needs to be added to the customer specific database. This should be used to help in the logic on how a dispatcher decides to deploy its resources to complete work in a work order.
9. Operator shifts and breaks should be incorporated into the model to more accurately represent the real world system.
10. Downtime needs to be added to the resources.

References

1. Walden, C.T. “Simulation Based Shipyard Planning System at Halter Marine.” Presentation at NSRP Joint Panel Meeting, San Diego, CA, June 2009.
2. Hill, T. W., Greenwood, A.G., Holt, R.E. & Walden, C.T., “Simulation Modeling: Obstacles Faced by Small and Medium Manufacturing Enterprises.” 2011 IERC Conference, proceedings of, Reno, NV, May 2011.
3. Hill, T.W. & Holt, R.E., “Linkage Tool Requirements.” December 31, 2010.
4. Albert Horsmon, Scott Clapham. Marine Systems Division Transportation Research Institute The University of Michigan. Standards Database Maintenance Phase II. Virginia: Newport News Shipbuilding Company; September 1997. NSRP 0488. P2479T-0-K1. 36 p.
5. Hill, T.W. & Holt, R.E., “Fabrication Data Requirements,” May 27, 2010.
6. Hill, T.W. & Holt, R.E., “Outfitting Data Requirements,” May 27, 2010.

APPENDIX A: System Requirements

The following hardware and software are required to use the PDS Tool and the prototype outfitting simulation model.

Hardware (Recommended):

CPU: Any modern Intel or AMD processor

RAM: 3 GB or more

Graphics: NVIDIA or ATI Graphics

OS: Windows XP, Vista, or 7

Hard Disk: 10 GB

Software:

Flexsim 5.1.2 or later

Microsoft Access 2003 or later

ShipConstructor 2009

APPENDIX B: Pilot Design Simulation Tool Usage

In the simulation model:

1. To use the Transfer Library within *Flexsim*, a modeler must toggle a code node as DLL and enter the following line where <PSD Tool Directory> is the current directory for the PSD Tool:

```
dll: "<PSD Tool Directory>\dataTransfer.dll" function: "TansferData"
```

2. When the code node gets executed the TransferData method within the Transfer Library will be ran.

In the Administrative GUI:

1. On the “Configuration” tab, setup the database configuration. You will need the following information for *ShipConstructor*:
 - Microsoft SQL Server and Instance Name
 - Username
 - Password
2. On the “Project” tab, select the projects you wish to include.
3. On the “Data” tab, press the “Update” to start the data extraction process (this may take several minutes or hours depending on the projects selected).

APPENDIX C: Intermediate Database XML Schema

```

<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="iDB">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ship">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="attr">
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base="xs:string">
                      <xs:attribute name="name" type="xs:string" use="required" />
                      <xs:attribute name="type" type="xs:string" use="required" />
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
              <xs:element maxOccurs="unbounded" name="unit">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element maxOccurs="unbounded" name="attr">
                      <xs:complexType>
                        <xs:simpleContent>
                          <xs:extension base="xs:string">
                            <xs:attribute name="name" type="xs:string" use="required" />
                            <xs:attribute name="type" type="xs:string" use="required" />
                          </xs:extension>
                        </xs:simpleContent>
                      </xs:complexType>
                    </xs:element>
                    <xs:element maxOccurs="unbounded" name="assembly">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element maxOccurs="unbounded" name="attr">
                            <xs:complexType>
                              <xs:simpleContent>
                                <xs:extension base="xs:string">
                                  <xs:attribute name="name" type="xs:string" use="required" />
                                  <xs:attribute name="type" type="xs:string" use="required" />
                                </xs:extension>
                              </xs:simpleContent>
                            </xs:complexType>
                          </xs:element>
                          <xs:element maxOccurs="unbounded" name="equipment">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element maxOccurs="unbounded" name="attr">
                                  <xs:complexType>
                                    <xs:simpleContent>
                                      <xs:extension base="xs:string">
                                        <xs:attribute name="name" type="xs:string" use="required" />
                                        <xs:attribute name="type" type="xs:string" use="required" />
                                      </xs:extension>
                                    </xs:simpleContent>
                                  </xs:complexType>
                                </xs:element>
                                <xs:sequence>
                                  <xs:attribute name="name" type="xs:string" use="required" />
                                </xs:sequence>
                              </xs:complexType>
                            </xs:element>
                          <xs:element maxOccurs="unbounded" name="extrusion">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element maxOccurs="unbounded" name="attr">

```

```

        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="name" type="xs:string" use="required" />
              <xs:attribute name="type" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element maxOccurs="unbounded" name="curvedplate">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="attr">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="name" type="xs:string" use="required" />
              <xs:attribute name="type" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element maxOccurs="unbounded" name="corrugationpart">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="attr">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="name" type="xs:string" use="required" />
              <xs:attribute name="type" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element maxOccurs="unbounded" name="platepart">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="attr">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="name" type="xs:string" use="required" />
              <xs:attribute name="type" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element maxOccurs="unbounded" name="flangepart">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="attr">
        <xs:complexType>

```

```

        <xs:simpleContent>
          <xs:extension base="xs:string">
            <xs:attribute name="name" type="xs:string" use="required" />
            <xs:attribute name="type" type="xs:string" use="required" />
          </xs:extension>
        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
</xs:complexType>
</xs:element>
<xs:element maxOccurs="unbounded" name="twistedextrusion">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="attr">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="name" type="xs:string" use="required" />
              <xs:attribute name="type" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element maxOccurs="unbounded" name="SWBS">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="attr">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:unsignedByte">
              <xs:attribute name="name" type="xs:string" use="required" />
              <xs:attribute name="type" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element minOccurs="0" name="pipe">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="attr">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="name" type="xs:string" use="required" />
              <xs:attribute name="type" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element minOccurs="0" name="hvac">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="attr">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="name" type="xs:string" use="required" />
              <xs:attribute name="type" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```



```

        </xs:simpleContent>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="name" type="xs:unsignedShort" use="required" />
</xs:complexType>
</xs:element>
<xs:element maxOccurs="unbounded" name="subassembly">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="attr">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="name" type="xs:string" use="required" />
              <xs:attribute name="type" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
      <xs:element maxOccurs="unbounded" name="extrusion">
        <xs:complexType>
          <xs:sequence>
            <xs:element maxOccurs="unbounded" name="attr">
              <xs:complexType>
                <xs:simpleContent>
                  <xs:extension base="xs:string">
                    <xs:attribute name="name" type="xs:string" use="required" />
                    <xs:attribute name="type" type="xs:string" use="required" />
                  </xs:extension>
                </xs:simpleContent>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attribute name="name" type="xs:string" use="required" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element maxOccurs="unbounded" name="curvedplate">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="attr">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="name" type="xs:string" use="required" />
              <xs:attribute name="type" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element maxOccurs="unbounded" name="corrugationpart">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="attr">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="name" type="xs:string" use="required" />
              <xs:attribute name="type" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

```

```

        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element maxOccurs="unbounded" name="platepart">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="attr">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="name" type="xs:string" use="required" />
              <xs:attribute name="type" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element maxOccurs="unbounded" name="flangepart">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="attr">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="name" type="xs:string" use="required" />
              <xs:attribute name="type" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element maxOccurs="unbounded" name="twistedextrusion">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="attr">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="name" type="xs:string" use="required" />
              <xs:attribute name="type" type="xs:string" use="required" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="name" type="xs:string" use="required" />
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="name" type="xs:string" use="required" />
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="name" type="xs:string" use="required" />
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="name" type="xs:string" use="required" />
</xs:complexType>
</xs:element>

```

```
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required" />
    </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="lastupdate" type="xs:string" use="required" />
</xs:complexType>
</xs:element>
</xs:schema>
```