DEACTIVATION DIAGRAM DEVELOPMENT FOR NAVAL SHIP SYSTEM VULNERABILITY ANALYSIS

Daniel Joseph Snyder

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
In
Ocean Engineering

Alan J. Brown, Chair
Julie Chalfant
Stefano Brizzolara

May 13th, 2019
Blacksburg, Virginia

Keywords: deactivation analysis, vulnerability, ship system design, architecture framework, distributed systems

DEACTIVATION DIAGRAM DEVELOPMENT FOR NAVAL SHIP SYSTEM VULNERABILITY ANALYSIS

Daniel Joseph Snyder

ACADEMIC ABSTRACT

System architecture analyses of distributed ship systems offer a practical view of system behavior over all operational states; however, the effectiveness of these analyses can be bound by limited computational performance or capability. Deactivation diagrams provide an alternative view to conventional system architecture descriptions, allowing for rapid analysis of system connectivity and flow based on precomputed single-state system descriptions. This thesis explores the development of system deactivation diagrams and their use in early-stage naval ship system design. Software tools developed in C++ and VBA as part of this research support the Virginia Tech (VT) Naval Ship Design Concept and Requirements Exploration (C&RE) process and tools utilizing the U.S. Navy's Leading-Edge Architecture for Prototyping Systems (LEAPS) framework database. These tools incorporate automated path-finding algorithms developed based on proven network theory and effective computational methods for use in performing ship system deactivation analysis. Data drawn from the results of this approach possess extensible applicability towards studies in naval ship system vulnerability, flow optimization, network architecture, and other system analyses. Supplementary work on interfacing the LEAPS framework libraries with deactivation analyses has demonstrated the capability for generating deactivation diagrams from complex LEAPS ship system databases and paved the way for future incorporation of LEAPS into research work at Virginia Tech.

DEACTIVATION DIAGRAM DEVELOPMENT FOR NAVAL SHIP SYSTEM VULNERABILITY ANALYSIS

Daniel Joseph Snyder

PUBLIC ABSTRACT

As the development of new ships becomes more technically complex due to the increased incorporation of redundant and interdependent ship systems, there is a greater need for advanced tools to support future ship system design. Ship operational capabilities rely on the resiliency of onboard systems in all situations, included damaged conditions, and require comprehensive design evaluation to identify weaknesses in system concepts. This thesis details the development of a computational approach to ship system analysis using precomputed deactivation diagrams for early-stage naval ship system design. Deactivation diagrams are a unique way of looking at the interconnectivity of system components and offer a consolidated view of complex network architecture to significantly simplify and accelerate subsequent analyses. Developments in computational algorithms for ship system connectivity presented in this thesis aid in the automated development of deactivation diagrams and support system flow and vulnerability analyses with particular regard to ongoing work on the Virginia Tech (VT) Naval Ship Design Concept and Requirements Exploration (C&RE) process. Additional thesis development work referencing the U.S. Navy's Leading-Edge Architecture for Prototyping Systems (LEAPS) database framework has demonstrated the capability for generating deactivation diagrams from complex LEAPS ship system databases and paved the way for future incorporation of LEAPS into research work at VT.

ACKNOWLEDGEMENTS

To my thesis advisor, Dr. Alan Brown, and my committee members, Dr. Julie Chalfant and Dr. Stefano Brizzolara, I wish to express my deepest gratitude for guiding me through this thesis and for all of the hours that you put in to help me see it through. Thank you for all of your time that you shared with me and for providing feedback and encouragement when I needed it most.

To my wonderful family, friends, and girlfriend, thank you for all of your support and understanding while I followed through on my goal to complete this work. Whenever I was discouraged, you were there for me, and whenever I had to postpone time spent with you, you gave me your patience. Thank you for all the love that you have shown me over these past few years.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Since the early days of ship design, ship systems have continued to grow in capability, complexity, and interdependency. With significant advances being made in decentralized power and control systems, the U.S. Navy has made the use of distributed systems a priority in new vessel design. At the 1989 American Society of Naval Engineers (ASNE) Day Conference, Chief of Naval Operations (CNO) Admiral Carlisle Trost stated that "…Integrated Electric Drive, with its associated cluster of technologies, will be the method of propulsion for the next class of surface battle force combatants,"[1] signaling development that would pave the way for modern Zonal Electrical Distribution Systems (ZEDS) and Integrated Power Systems (IPS), such as those introduced in the U.S. Navy Surface Fleet aboard the USS Zumwalt (DDG1000) destroyer class.

Distributed systems provide significantly increased flexibility and capability over their predecessors at the cost of system simplicity and repairability. In addition to improved operational effectiveness, the decentralization of critical system components has made it possible to improve upon overall vulnerability by the deliberate design of distributed redundant systems throughout a ship's hull. Despite these benefits, the study of ship vulnerability to damage, commonly known as vulnerability analysis, remains a complex process and a hinderance to ship design. Due to this, such analyses are often largely deferred from the concept design phase and may ultimately have little to no impact on ship arrangements and system selection when completed during preliminary or detail design phases.

One approach to enhancing the throughput and capability of vulnerability analyses is through the development of deactivation diagrams. These diagrams provide an alternative view of the ship system architecture by describing all of the system connections in an explicit and unidirectional layout. Distributed ship systems often contain connections between system components that may operate with the system commodity "flow" in either direction through the same conduit based on the system state, referred to as bidirectional connectivity. The transformation of a system architectural view from a bidirectionally-connected description to the unidirectional deactivation diagram can have significant computational complexities; however, the precomputed deactivation structure yields significant advances to subsequent analyses based on its straightforward design.

This thesis is intended to provide a foundational mechanism for ship system deactivation analysis by constructing a methodology and tools for the logical development of deactivation diagrams. These efforts seek to support developments in vulnerability analyses proposed by Goodfriend [2] and provide added value to other ongoing efforts in architectural analysis [3] and flow optimization [4] at Virginia Tech (VT).

The design methodology and tools developed in this thesis are targeted for use in early stage naval ship design using multi-objective optimization routines currently used in ship design courses taught at VT. By developing the framework for rapid deactivation analysis during concept design, useful information on system architecture and survivability may be derived and influence early ship design choices. These choices may have a significant impact on the ship design path and promote alternative choices that better support the ship's mission. The work completed in this thesis builds upon the work of others [2, 5, 6] to further develop the concept and capabilities of the

VT Concept and Requirements Exploration (C&RE) process and the Ship Synthesis Module (SSM) for performing domain-based concept design.

1.1 Objectives

The objectives of this thesis include presentation of a methodology for creating deactivation diagrams from system architecture descriptions and development of tools useful for continued research into ship concept design and vulnerability analysis. The first part of this effort is to develop a robust and practical code for deactivation diagram development targeting work on ship systems currently ongoing at Virginia Tech and the second part is to assemble an intermediary tool for effective LEAPS database interface with the deactivation diagram tool.

1.1.1 Deactivation Diagram Development

In order to meet the current capability gap in concept design vulnerability analysis, prior work completed by David B. Goodfriend [7] delved into the issue of rapid ship system vulnerability analysis for naval combatant design. This approach to vulnerability analysis requires explicit unidirectional connectivity between system components presented in the deactivation diagram. To develop deactivation diagrams for use in vulnerability analysis, Mr. Goodfriend's work utilized external system input provided by ITEM ToolKit, a commercial reliability analysis software package. This thesis provides an alternative approach to deactivation analysis and deactivation diagram development without the dependence on external tools or manual system analysis. An example of a deactivation diagram developed using the methods detailed in this thesis is shown in Figure 1.1.

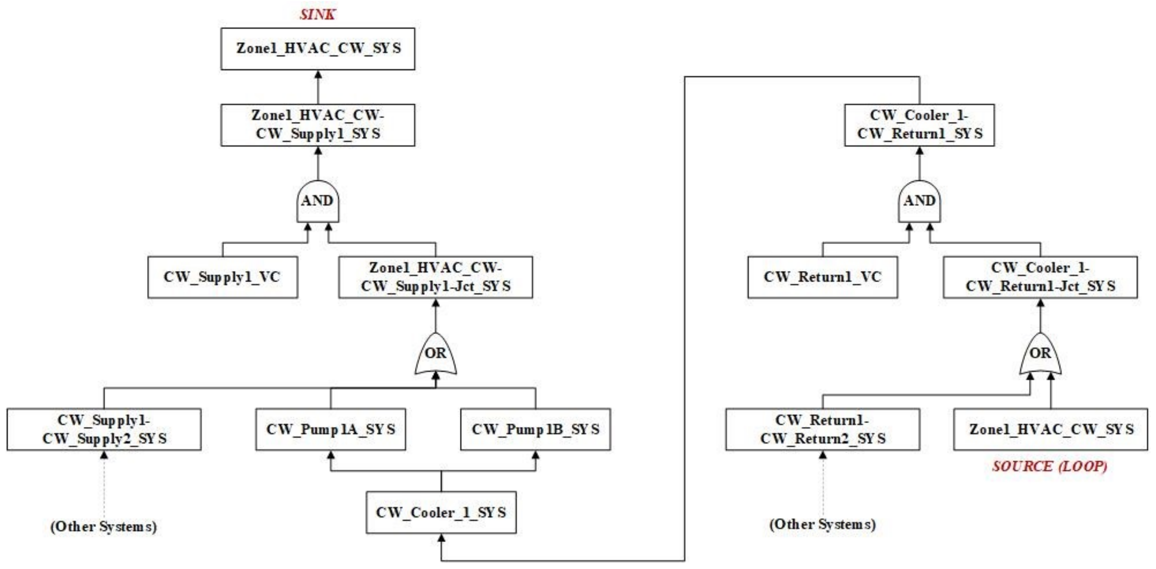

SINK
Zone1_HVAC_CW_SYS
Zone1_HVAC_CW-CW_Supply1_SYS
AND
CW_Supply1_VC
Zone1_HVAC_CW-CW_Supply1-Jct_SYS
OR
CW_Supply1-CW_Supply2_SYS
CW_Pump1A_SYS
CW_Pump1B_SYS
CW_Cooler_1_SYS
(Other Systems)
CW_Cooler_1-CW_Return1_SYS
AND
CW_Return1_VC
CW_Cooler_1-CW_Return1-Jct_SYS
OR
CW_Return1-CW_Return2_SYS
Zone1_HVAC_CW_SYS
SOURCE (LOOP)
(Other Systems)

Figure 1.1: Example Chilled Water System Deactivation Diagram

Using Microsoft's Visual Basic for Applications (VBA) software development framework included in Microsoft Excel, this thesis will demonstrate the design and use of a modular tool built for connectivity path analysis and deactivation diagram development based on a set of practical input data. This tool is easy to adapt and maintain through the selective use of VBA, yet it has been highly optimized and built with numerous safety checks and data validation algorithms for robustness and effectiveness during practical use.

1.1.2 LEAPS Integration

As the complexities and costs of distributed ship systems continue to grow along with the greater availability of multi-mission capable options, designers are seeking new approaches for automated analysis of new concepts in bulk. To meet the demands for optimal system selection and budgetary constraints, the development of ship system descriptions at preliminary stage is being promoted for future naval ship design and new tools will be needed to run optimization and selection routines.

The U.S. Navy is currently working in conjunction with research groups and academic institutions on the development of tools to aid in computer-aided ship naval ship design and are pushing these toolsets as the de-facto standard. These tools, including the Leading-Edge Architecture for Prototyping Systems (LEAPS) data repository and Smart Ship System Design (S3D) tool suite, have been designed to capture multiple ship concept and ship system designs in a large database for architectural analysis. LEAPS provides a foundation for the digital representation of all ship systems and components, including explicit system connectivity and capacities, as shown in Figure 1.2. Work on the VT C&RE process targets similar design processes and provides a strong basis for the future incorporation of LEAPS data.

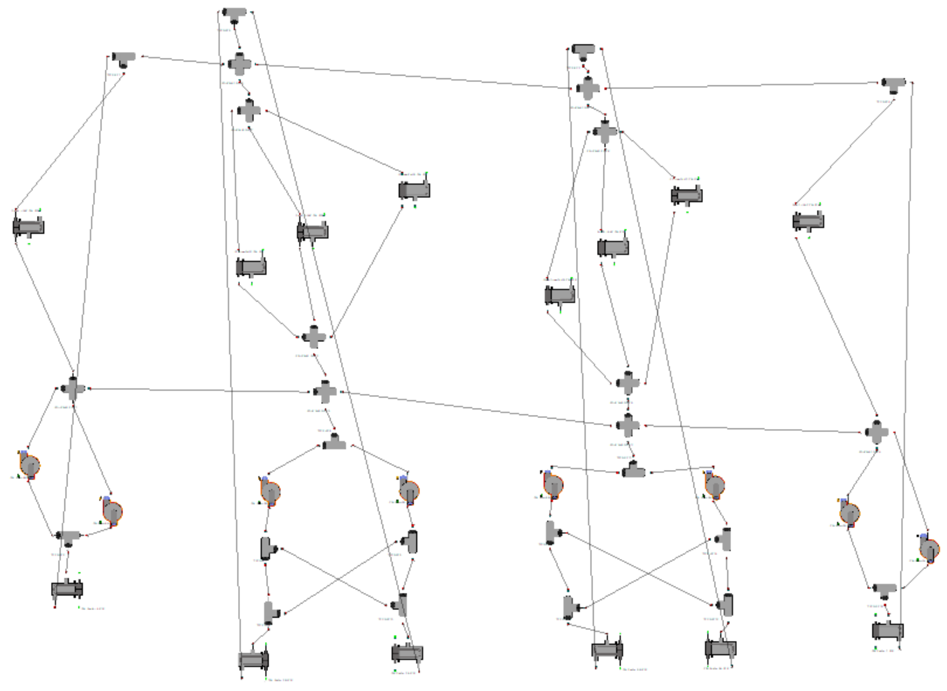


Figure 1.2: Example Chilled Water System Shown in S3D

This thesis seeks to enhance the state-of-the-art of the VT SSM through the integration of the deactivation diagram tool with LEAPS databases in the LEAPS Network Translator tool. The standalone tool developed for this effort promises to extract the system component definitions necessary to completely describe the various ship systems needed for vulnerability analysis. This extracted information, including component names, descriptions, and connectivity, build the total system description necessary to run the deactivation diagram tool and generate system-specific deactivation diagrams for the ship concept and for use in VT ship design studies.

1.2 Manuscripts

This thesis is presented in the VT manuscript format, including two co-authored manuscripts detailing work led and contributed to by the author in fulfillment of the thesis requirement. The two manuscripts presented in this thesis were prepared in collaboration with academic co-authors and derived from research performed at VT, Massachusetts Institute of Technology (MIT), and by the Naval International Cooperative Opportunities in Science and Technology Program (NICOP).

The author of this thesis is the lead author of both papers and the primary developer of applications described in both manuscripts. These following manuscripts have been prepared for submission and publishing:

- Network Architecture Framework Applications with FOCUS-Compliant Ship Designs
    - Co-Authored with Mark A. Parsons, Julie Chalfant, and Alan J. Brown
- Naval Ship System Deactivation Analysis Using Network Architecture Framework
    - Co-Authored with Mustafa Y. Kara, Mark A. Parsons, and Alan J. Brown

1.3 References

[1] C. F. Krolick and C. Graham, "Technology Clusters," in *Naval Engineers Journal*, 1989, pp. 27-33.
[2] D. B. Goodfriend, "Exploration of System Vulnerability in Naval Ship Concept Design," M.S. Thesis, Department of Aerospace and Ocean Engineering, Virginia Polytechnic and State University, Blacksburg, VA, 2015.
[3] M. A. Parsons *et al.*, "Application of a Distributed System Architecture Framework to Naval Ship Concept and Requirements Exploration (C&RE)," in *Proc. ASNE Intelligent Ship Symposium*, 2019, in press.
[4] M. A. Parsons *et al.*, "Early-Stage Naval Ship Distributed System Design Using Architecture Flow Optimization," *Naval Engineers Journal*, unpublished.
[5] A. J. Brown and J. Salcedo, "Multiple-Objective Optimization in Naval Ship Design," *Naval Engineers Journal,* vol. 115, no. 4, pp. 49-62, 2003.
[6] A. J. Brown and J. Waltham-Sajdak, "Still Reengineering the Naval Ship Concept Design Process," *Naval Engineers Journal,* vol. 127, no. 1, pp. 49-61, 2015.
[7] D. B. Goodfriend, "Exploration of System Vulnerability in Naval Ship Concept Design," Virginia Polytechnic Institute and State University, Blacksburg, VA, 2015.

Chapter 2

Network Architecture Framework Applications with FOCUS-Compliant Ship Designs

(Submitted to the 2019 IEEE Electric Ship Technologies Symposium)

Daniel J. Snyder, Mark A. Parsons, and Alan J. Brown
Kevin T. Crofton Dept. of Aerospace and Ocean Engineering
Virginia Tech
Blacksburg, VA, USA

Julie Chalfant
MIT Sea Grant College Program
Massachusetts Institute of Technology
Cambridge, MA, USA

Abstract

Ongoing development of the US Navy's LEAPS data repository, research exploring the use of networks in ship system design by the Naval International Cooperative Opportunities in Science and Technology Program (NICOP), and tool development by the Electric Ship Research and Development Consortium (ESRDC) have provided an excellent opportunity to interface efforts to address preliminary distributed system design and analysis in early-stage ship design. This paper builds on the architecture framework and network-based methods by demonstrating new ways to process FOCUS-compliant data, develop network views, and analyze connectivity and flow of distributed ship systems. This framework decomposes system architecture into three primary views: physical, logical, and operational. Network-based tools targeted on these views and their intersections efficiently explore and analyze broad ranges in the ship system design space. These explorations generate knowledge, encourage innovation, and support the synthesis of affordable, effective ship designs. The authors present the findings of these efforts in consideration for future changes to the FOCUS Product Meta-Model to support integrated advanced network architecture analyses.

Keywords — *Architecture Framework, LEAPS, Network Theory, Ship System Design*

This work is supported by the Office of Naval Research (ONR) under ONR N00014-15-1-2476 (NICOP) Naval Ship Distributed System Vulnerability and Battle Damage Recovery in Early-Stage Ship Design, US Naval Surface Warfare Center Philadelphia N00174-16-C-0048 Network-Based System Architecture Assessment and Improvement in Support of S3D/LEAPS Ship System Design, and ONR N00014-16-1-2945 Incorporating Distributed Systems in Early-Stage Set-Based Design of Navy Ships. Approved for public release under DCN# 43-5021-19.

2.1 Introduction

In the naval ship design community, there exists a critical need to expand the theory and capabilities of early-stage ship distributed system design tools. Addressing this need requires fundamental understanding of foundational systems engineering, an appreciation for the current status of existing tools and ongoing research, and new ideas. The System Architecture Framework described in Section 2.2 is a novel system network decomposition approach that offers great promise for understanding and designing ship distributed systems. Section 2.3 expands on this framework description and introduces the concept of operational nodes denoting system operations not associated with a single component. Examples of these ideas are provided in later sections using prototyped tools to demonstrate their underlining theory.

Several of these tools create and/or use network descriptions of systems for ship design and analysis. An example of this is the Architecture Flow Optimization (AFO) method described in Section 2.4. This method accomplishes a linear optimization of energy flows within ship systems in the early stages of ship design to minimize flow cost, ensure operational capabilities are satisfied, and reduce system vulnerability.

The U.S. Navy has developed a standard data repository, the Leading-Edge Architecture for Prototyping Systems (LEAPS) [1], with the goal of providing a single, consistent, reliable framework for storing design data for all Navy design tools. Section 2.5 discusses the method of storing system data in the LEAPS repository and describes custom software tools under development supporting the extraction and processing of network architecture from external LEAPS databases in order to develop directed graphs and export data to other tools for further post-processing. Based on experience derived from the development of these tools, the authors also present some of the ways that LEAPS and the Formal Object Classification for Understanding Ships (FOCUS) Product Meta-Model (PMM) may be subsequently modified to further support future network theory analyses.

Section 2.6 describes specific work on a tool developed by Virginia Tech (VT) which transforms the archetypical system data from a singular-flow model into system deactivation diagram(s), providing instantaneous analysis of multiple operational flow patterns in distributed systems. This type of system view is commonly used for system availability and vulnerability analysis. An abbreviated theory and design description of the tool accompany the detail of its use with examples developed from the AFO flow model and the VT Concept & Requirements Exploration (C&RE) example ship system model.

In summary, this paper describes and demonstrates methodologies for analyzing naval ship distributed systems, defining required components for operational capabilities, validating early-stage naval ship distributed systems designs, manipulating FOCUS-compliant ship data, creating deactivation diagrams, and proposes additions to the FOCUS PMM and LEAPS tools.

2.2 Architectural Framework

Brefort et al [2] describes system architecture as decomposable into physical, logical, and operational views by the architecture framework for distributed naval ship systems, as shown in Figure 2.1. According to Brefort et al:

> This representation describes the spatial and functional relationships of the system together with their temporal behavior characteristics.[…]The physical architecture describes the spatial arrangement, the logical architecture describes information on the functional

characteristics of the system, and the operational architecture contains information on the temporal behavioral characteristics of the vessel, in a given mission scenario. [2, pp. 375-377]

This framework enables individual architectures to be implemented separately as illustrated in Figure 2.2 and integrated to solve for physical solutions, physical behavior, functional utilization and ultimately the total system response. Being able to access LEAPS component data views applicable to this network framework is a first step to a network theory implementation.

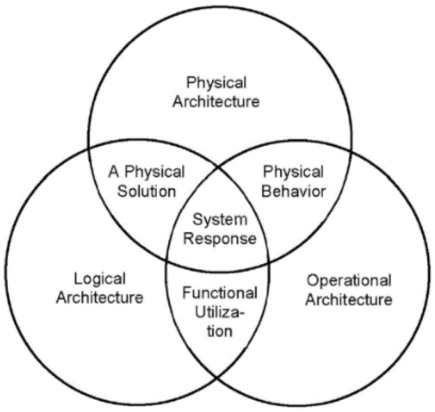


Figure 2.1: Representation of an Architectural Framework for Ship Distributed Systems [2]

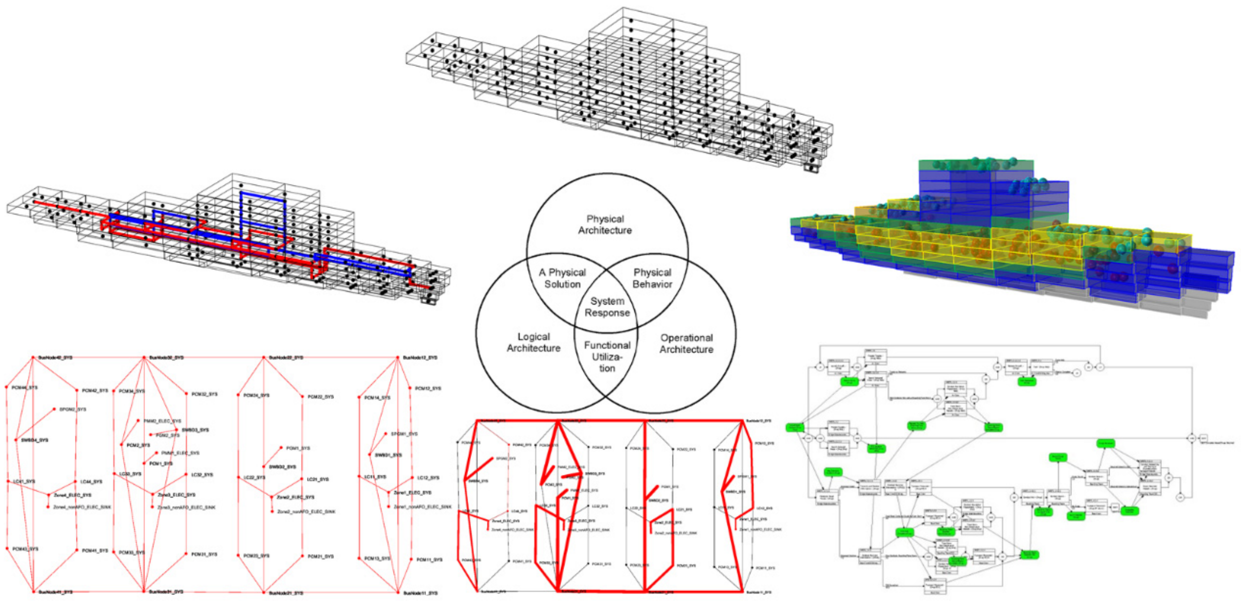


Figure 2.2: Notional Architecture Framework Implementation in Ship Concept Exploration [3]

The network logical architecture is the most fundamental aspect of the architecture framework. Figure 2.3 shows a simple logical system architecture of a mechanical subsystem or plex of a larger integrated power system. This network representation is made up of nodes and the edges that connect them, in which each node is either a vital component ("VC") or system node ("SYS"). System nodes may represent sources, sinks or ports of vital components in other plexes. Each VC in this architecture also has physical attributes and is ultimately located physically in the ship in the "physical solution."

The Propulsion_SYS node shown in Figure 2.3 is an operational system node that provides propulsion capability to the operational architecture and pulls energy from the mechanical energy plex and ultimately the entire ship system multiplex to support a required level of performance, in this case meeting a designated ship propulsion speed. In order to operate, the mechanical subsystem shown in Figure 2.3 requires functional capability of other plexes within the multiplex system, including electric power, machinery control, lube oil, HVAC, seawater, chilled water, and HFC. When viewed in a multiplex deactivation diagram, these systems define the total ship propulsion system as shown in Figure 2.10.

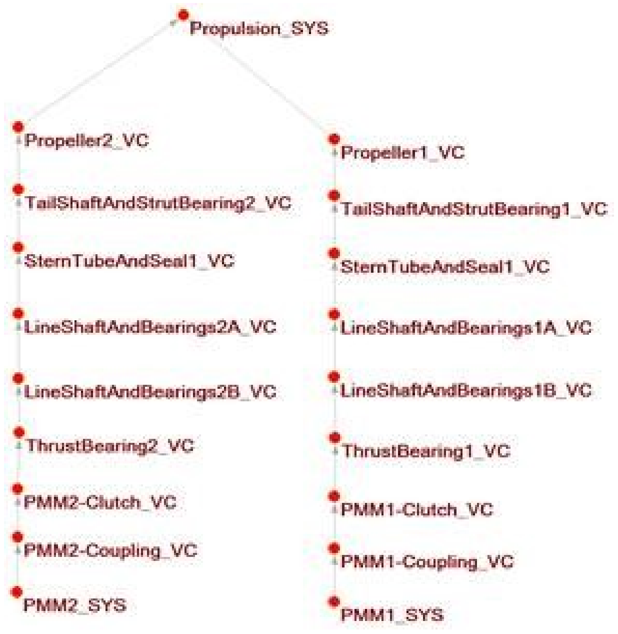


Figure 2.3: Mechanical (Propulsion) Subsystem or Plex Logical Architecture
(PMM in this figure stands for Propulsion Motor Module)

2.3 Operational Systems

A distribution system is defined as: the group of components and their connections whose purpose is to distribute a commodity (e.g. physical substance, energy, materiel, or information) from sources to sinks [4]. This commodity-based definition of a system is the default definition many marine engineers use. A perfect example of a system following this definition is a traditional chilled water system, which includes any components the chilled water commodity flows through.

An alternate operation-based definition could be: the components and logical connections required to transport energy, carried by other commodities from sources to a single operational sink, including any redundant paths. Operations produce, transform, or consume energy. These operations may interact with the ship's environment or other operational systems within the ship. If an operational sink is also the capability node in a deactivation diagram, this node and the remaining components and their connections in the deactivation diagram form its operational system. This alternate definition was conceived as an important addendum to the operational architecture view established by Brefort et al. [2].

Figure 2.3 shows an example of a mechanical propulsion system (mechanical energy distribution system) with an additional operational system node connecting the two shafts. Both propellers connect to this Propulsion_SYS node. This node represents the energy sink of the system, and its demand value is determined by the operational condition of the ship (e.g. sustained speed, endurance speed, battle speed, etc.). Operational conditions may affect which parts of the system are classified as vital or redundant. For example, the sustained speed condition may require both shafts, while the endurance speed condition only requires a single shaft. For this reason, it is important to have a fully defined operational architecture including design reference missions when evaluating marine engineering systems (especially mission systems) [2-4].

Figure 2.3 is both an operational system and deactivation diagram when only considering the mechanical components in the propulsion system. The system can be expanded to include all necessary ship system components as shown in Figure 2.10. In this view, the propulsion motor modules require electrical power and lube oil cooling. The power generation modules that provide electrical power require fuel oil. The electrical components and lube oil pumps are cooled by chilled water components. Finally, the lube oil and chilled water components are cooled by seawater. All of these components are part of the total propulsion operational system.

2.4 Architecture Flow Optimization

The network architecture framework facilitates useful applications such as Architecture Flow Optimization (AFO) or linear energy flow optimization. In this approach, complex behaviors like pump curves, engine maps, power conversion, or heat exchange are modeled by simple energy flow coefficients and enforcing conservation of energy at each node. These coefficients are unique to each component type and are stored in a comprehensive Machinery Equipment List (MEL) for the total system. *Through variables* (e.g. current, flow rate, speed) or *cross variables* (e.g. voltage, pressure, torque) are not used.

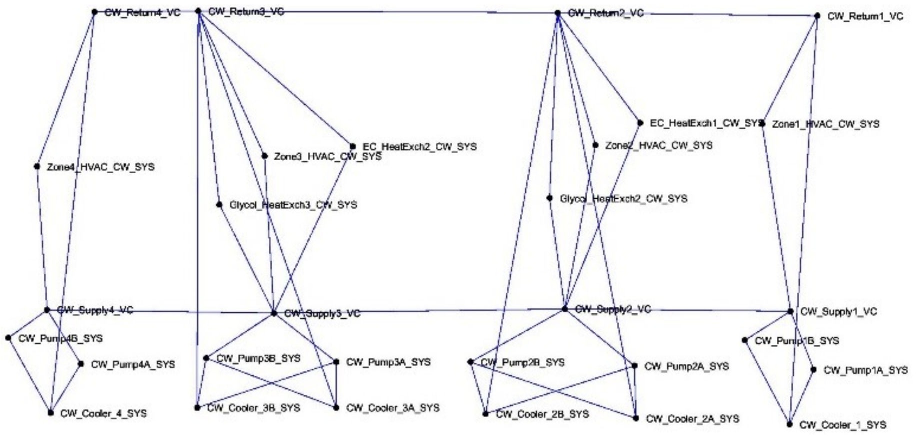


Figure 2.4: AFO Chilled Water Plex Logical Architecture

The AFO begins with a system logical architecture (example shown in Figure 2.4), applies these energy coefficients in the MEL, and enforces steady-state or quasi-steady-state operational constraints. The objective function in this optimization minimizes the flow cost of the network. This cost has two components: a fixed cost (representing the engineering and instillation costs of connecting components) and a variable cost (which linearly increases with flow). Parsons et al. [5] and Brown [4] provide a complete description of the linear optimization formulation and the operational constraints.

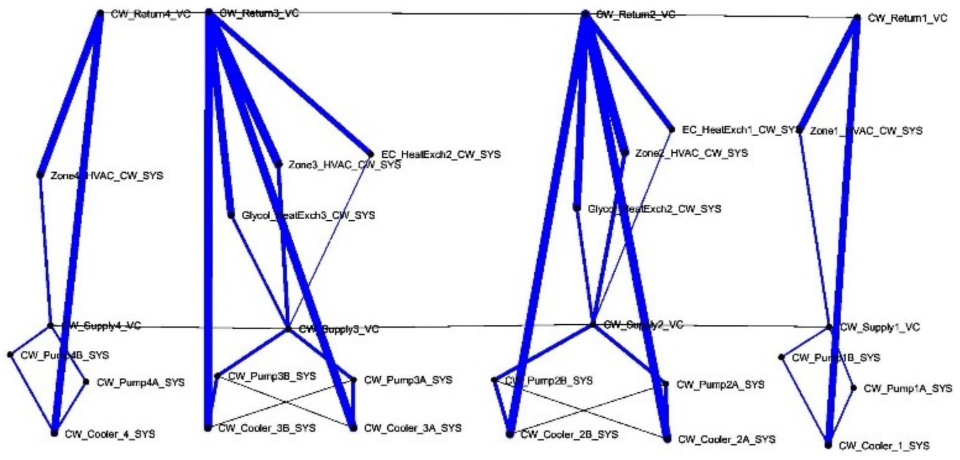


Figure 2.5: AFO Chilled Water Plex Sustained Speed Functional Utilization

Figure 2.5 shows the sustained-speed condition functional utilization of the AFO for the chilled water example provided in Figure 2.4. The functional utilization is the intersection of the logical and operational architectures shown in Figure 2.1 and represents the time-domain utilization of the system. These energy flows are used to parametrically size the volume, area, and weight of components. Brown [4] states that the application of the network architecture framework in conjunction with this energy flow method has a number of significant advantages:

1. Explicit sizing of major combat, power, and energy components, early in the design process.
2. Consideration of broader ranges of system options and architectures outside of the range of historical data-based parametrics.
3. Enables early preliminary arrangements.
4. Enables early distributed system architecture optimization.
5. Enables a more specific consideration of operational architecture scenarios including warfighting damage.
6. Enables early flow-based maintenance, reliability and availability analyses which,
7. Enables early vulnerability and recoverability analyses.
8. Excellent tool for understanding and communicating energy flow through a complex distributed system of systems.

This energy flow method serves as a potential low fidelity (in terms of both physics and number of components) but sufficient method for concept exploration, initial equipment sizing, and system validation. Tools like S3D have a stronger physics-based commodity flow solver and are better suited to analyzing more detailed/complete system designs with piping, ducting, and other commodity distribution components [3].

2.5 LEAPS Integration

The U.S. Navy's LEAPS data repository has been developed to provide a consistent and reliable framework for storing design data. Within a LEAPS database, the data structure of surface ship-related data is strictly controlled by the Formal Object Classification for Understanding Ships (FOCUS) product meta-model (PMM), which maintains the categorization of information according to predefined rules and object types. Additional data may be stored alongside FOCUS data in the database, but is without the organization and quality assurance of the FOCUS PMM.

Evolution of the PMM requires procedural updates and a predetermined need, further enforcing consistency rules on the storage and extraction of ship data.

Despite the robustness of the LEAPS database, certain gaps currently exist within the FOCUS framework for seamlessly storing and accessing component data views directly applicable to network theory. Utilizing the extensibility of the LEAPS framework, custom software codes such as those presented in this paper have been independently developed to parse and store additional information within the LEAPS database for network analysis. Ongoing goals of this work include developing the demonstratable benefits of seamless integration of network information into the FOCUS PMM for analytical purposes.

There are several design tools and methodologies which benefit from a robust method of storing system data in LEAPS and extracting system network diagrams from LEAPS. The AFO described in Section IV above is the primary example used in this paper. Some additional tools include the following:

Smart Ship Systems Design (S3D) [6] is a software framework that can be used to define, simulate and analyze shipboard distribution systems in the electrical, thermal and mechanical energy domains. Systems are constructed by selecting components from an equipment library, and arranging and connecting them in discipline-specific views. Power-flow-level simulation can then be conducted on the assembled system designs. The resultant systems are stored in a LEAPS database in a FOCUS-compliant manner.

The *System Builder* [7] software is a body of work with the goal of achieving semi-automated design of ship systems, in which the systems are generated automatically under the guidance of the engineer or designer, using a templating process. Templates in this application are pre-designed sections of systems stored in a LEAPS database; they can be created using the S3D software. The templates are assembled into fully connected, fully functioning ship systems with components placed in three dimensions in a ship hullform. One step of the templating process requires determining maximum power flow through each component making up a system; this enables sizing of the components both in terms of dimensioning and in terms of managing the power. This is achieved by extracting a network representation of the system from the LEAPS database and applying a maximum flow algorithm, described in [7] to that network.

The *LEAPS Network Translator* is the result of an ongoing collaboration between MIT and Virginia Tech (VT) to create merged software that provides straightforward network representations of systems for use in external clients such as the System Builder, AFO, and the VT Ship Synthesis Module (SSM). Utilizing the System Builder framework, additional capability was developed to isolate individual system networks and process source/sink data and graph directionality as required for deactivation diagram analysis. This tool extracts system descriptions from a LEAPS database developed in S3D and applies network theory analysis to the system using a system of recursive algorithms and pre-defined object characteristics to develop a network description exportable to tools currently under development at VT. Output from the tool is formatted to match the Pajek large network analysis software ".net" file format.

Owing to the use of LEAPS as a data repository, all of these tools can be used in conjunction with one another. Systems defined using S3D can be assessed using AFO and the LEAPS Network Translator; templates created in S3D can be assembled, sized and placed using System Builder then analyzed using S3D or AFO.

2.5.1 System Diagram Representation in LEAPS

Network theory commonly describes the layout and connections of a system in the form of an adjacency matrix or adjacency list, each uniquely representing a system using nodes (vertices) and edges. Each node is representative of a system component, being a unique vital component or a child sub-system for the parent system. Edges identify the interconnectivity between pairs of vertices and may represent either the direct connections between components or substitute for static distribution system components such as pipes and cables. Additionally, an edge may be identified as an undirected edge for bidirectional flow or a directed edge, also referred to as an arc, for unidirectional flow.

Within the LEAPS database, LEAPS Components take the place of system vertices and the edges are represented by LEAPS Exchange Connections. We note in passing that a LEAPS Node is not the same concept as the graph node discussed herein. Although LEAPS does not specifically store networks either as adjacency matrices or adjacency lists, all the information required to extract such representations is available. See [8] for more information on the specific manner in which a system is stored in LEAPS.

One of the features of LEAPS is that defined system components may be uniquely represented in multiple common views, systems, and diagrams in different manners. As such, a single component such as a water chiller can appear as a thermal source in a chilled water system diagram, a thermal load in a seawater system diagram, and an electrical load in an electrical system diagram.

The mechanical propulsion system shown in Figure 2.3 could be represented in LEAPS using FOCUS-compliant components for each VC listed and appropriate connection structure between the applicable nodes of those components. However, in order to represent the operational system functionality shown by the Propulsion_SYS top node and the two PMM_SYS nodes, a new type of LEAPS Component needs to be included in the FOCUS PMM.

At this point, neither LEAPS nor the FOCUS PMM include directionality information for connections. Such information can be parsed from the types of components and types of nodes associated with components, but a more robust method would be to include such information in the FOCUS definition of a LEAPS Terminal, which is a type of LEAPS Node. This is an important addition that is needed for network analysis of systems.

2.5.2 Component Analysis Using LEAPS Network Translator

Systems built using S3D are flexible in detail and are typically scoped to the level of detail required by the end user. Figure 2.6 shows the chilled water system chosen as a test case for the analysis demonstrated by this paper. Detail design elements, such as pipes and most pipe fittings, are superfluous to this analysis and are omitted from this demonstration model.

Contrasting with the flow analysis model shown in Figure 2.4, S3D constrains connections to the number of physical ports on each component. Figure 2.6 gives an example of how piping connectivity must be realistically modelled via multi-directional fittings or distribution manifolds.

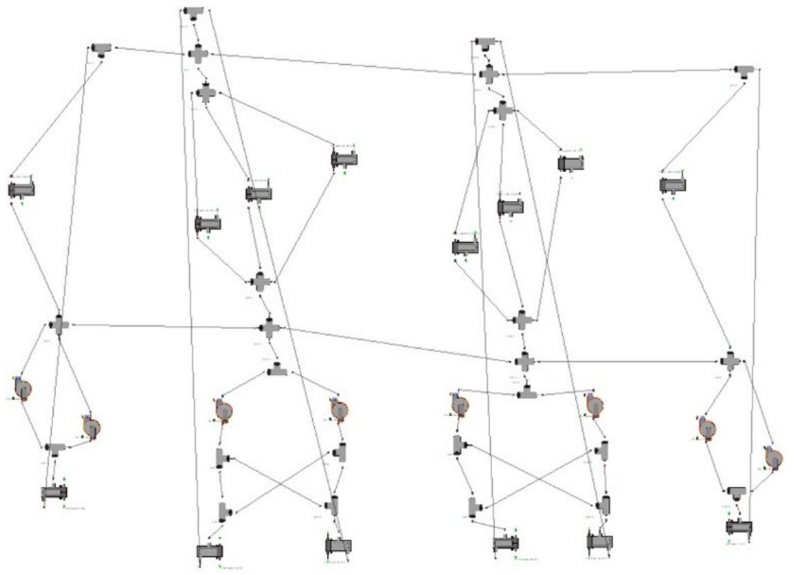


Figure 2.6: S3D Chilled Water Piping Schematic

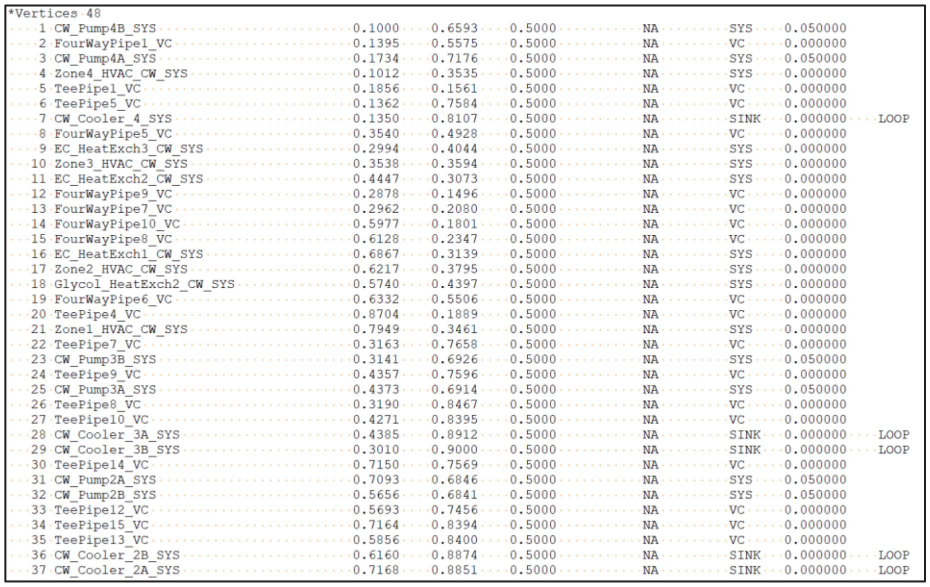
The chilled water system in Figure 2.6 demonstrates the physical layout and connections of the system without predefined flow patterns (determined by internal flow analysis). System Builder directionality tools for directed graphs are used to identify and cache fixed input and output nodes by comparing the name and id of each node to a set of carefully-selected criteria. The LEAPS Network Translator provides additional analysis tools and criteria for extracting the necessary connectivity information for building a deactivation diagram. These predetermined criteria include assumptions like uni-directional liquid flow through a typical pump and no power generation by devices not designed to do so.

System loads (sinks) are identified by a static list of component types and nodal properties built into the tool. Sinks in the chilled water piping schematic shown in Figure 2.6 are located as the hot water inlets in the chilled water heat exchangers. A depth-first search (DFS) algorithm is used to find the system sources as the independent elements farthest from the sinks in the adjacency matrix. Loop systems, where the system sink shows up in its own recursive analysis, are also identified and managed accordingly.

2.5.3 Data Export

During the development of the LEAPS Network Translator, the preferred export file format chosen for use by the VT SSM was the space-delimited Pajek ".net" text file format for its ease of maintenance and compatibility with most network graph tools (Pajek, Gephi, NodeXL, NetworkX, etc). This format allows for additional information to be stored with each vertex definition without affecting the basic operability of the software. This allows for expandability to include data such as S3D properties, network types, and external system connections.

The LEAPS Network Translator export data structure, shown in Figure 2.7, contains the vertex definition list and subsequent lists of corresponding arcs and edges. Most vertices are identified as a VC or SYS type in the output text. In each system description, at least one vertex is identified as a system sink ("SINK"), with additional data columns identifying either a system loop pattern or the corresponding network sources for a serial (non-loop) system. Figure 2.8 shows the Pajek ".net" format representation of the chilled water system from Figure 2.6.



```
*Vertices 48
    1 CW_Pump4B_SYS                     0.1000    0.6593    0.5000          NA         SYS    0.050000
    2 FourWayPipe1_VC                   0.1395    0.5575    0.5000          NA         VC     0.000000
    3 CW_Pump4A_SYS                     0.1734    0.7176    0.5000          NA         SYS    0.050000
    4 Zone4_HVAC_CW_SYS                 0.1012    0.3535    0.5000          NA         SYS    0.000000
    5 TeePipe1_VC                       0.1856    0.1561    0.5000          NA         VC     0.000000
    6 TeePipe5_VC                       0.1362    0.7584    0.5000          NA         VC     0.000000
    7 CW_Cooler_4_SYS                   0.1350    0.8107    0.5000          NA         SINK   0.000000    LOOP
    8 FourWayPipe5_VC                   0.3540    0.4928    0.5000          NA         VC     0.000000
    9 EC_HeatExch3_CW_SYS               0.2994    0.4044    0.5000          NA         SYS    0.000000
   10 Zone3_HVAC_CW_SYS                 0.3538    0.3594    0.5000          NA         SYS    0.000000
   11 EC_HeatExch2_CW_SYS               0.4447    0.3073    0.5000          NA         SYS    0.000000
   12 FourWayPipe9_VC                   0.2878    0.1496    0.5000          NA         VC     0.000000
   13 FourWayPipe7_VC                   0.2962    0.2080    0.5000          NA         VC     0.000000
   14 FourWayPipe10_VC                  0.5977    0.1801    0.5000          NA         VC     0.000000
   15 FourWayPipe8_VC                   0.6128    0.2347    0.5000          NA         VC     0.000000
   16 EC_HeatExch1_CW_SYS               0.6867    0.3139    0.5000          NA         SYS    0.000000
   17 Zone2_HVAC_CW_SYS                 0.6217    0.3795    0.5000          NA         SYS    0.000000
   18 Glycol_HeatExch2_CW_SYS           0.5740    0.4397    0.5000          NA         SYS    0.000000
   19 FourWayPipe6_VC                   0.6332    0.5506    0.5000          NA         VC     0.000000
   20 TeePipe4_VC                       0.8704    0.1889    0.5000          NA         VC     0.000000
   21 Zone1_HVAC_CW_SYS                 0.7949    0.3461    0.5000          NA         SYS    0.000000
   22 TeePipe7_VC                       0.3163    0.7658    0.5000          NA         VC     0.000000
   23 CW_Pump3B_SYS                     0.3141    0.6926    0.5000          NA         SYS    0.050000
   24 TeePipe9_VC                       0.4357    0.7596    0.5000          NA         VC     0.000000
   25 CW_Pump3A_SYS                     0.4373    0.6914    0.5000          NA         SYS    0.050000
   26 TeePipe8_VC                       0.3190    0.8467    0.5000          NA         VC     0.000000
   27 TeePipe10_VC                      0.4271    0.8395    0.5000          NA         VC     0.000000
   28 CW_Cooler_3A_SYS                  0.4385    0.8912    0.5000          NA         SINK   0.000000    LOOP
   29 CW_Cooler_3B_SYS                  0.3010    0.9000    0.5000          NA         SINK   0.000000    LOOP
   30 TeePipe14_VC                      0.7150    0.7569    0.5000          NA         VC     0.000000
   31 CW_Pump2A_SYS                     0.7093    0.6846    0.5000          NA         SYS    0.050000
   32 CW_Pump2B_SYS                     0.5656    0.6841    0.5000          NA         SYS    0.050000
   33 TeePipe12_VC                      0.5693    0.7456    0.5000          NA         VC     0.000000
   34 TeePipe15_VC                      0.7164    0.8394    0.5000          NA         VC     0.000000
   35 TeePipe13_VC                      0.5856    0.8400    0.5000          NA         VC     0.000000
   36 CW_Cooler_2B_SYS                  0.6160    0.8874    0.5000          NA         SINK   0.000000    LOOP
   37 CW_Cooler_2A_SYS                  0.7168    0.8851    0.5000          NA         SINK   0.000000    LOOP
```

Figure 2.7: LEAPS Network Translator Chilled Water Pajek Export Format

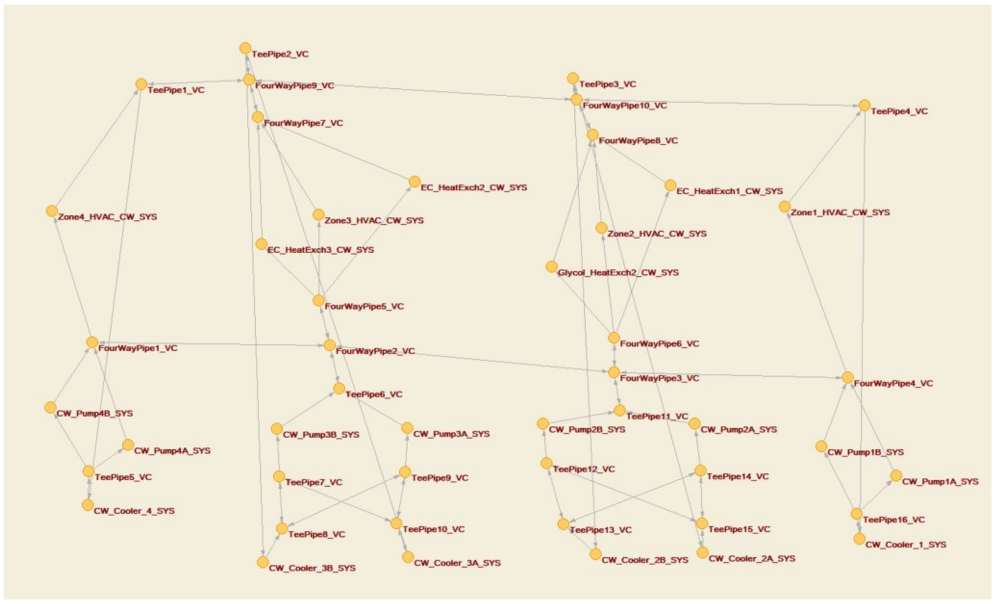


Figure 2.8: Visualization of S3D-based Chilled Water Schematic in Pajek

2.6 Deactivation Diagram Tool

Adjacency lists or matrices can be developed to represent and analyze most systems; however, they are often insufficient for efficient analysis of multi-directional flow distributed systems. To develop an effective connectivity study of multi-directional ship systems, such as the zonal electric distribution system (ZEDS), it is often helpful or required to treat all system components as unidirectional within the scope of the analysis for much greater throughput. Preprocessing of multi-path system flow by way of a deactivation diagram can offer simultaneous state analysis and significantly improve the performance and capability of other connectivity analyses.

One such example of simultaneous analysis based on deactivation diagram unidirectional connectivity is the rapid ship system vulnerability analysis developed by Goodfriend [9] for incorporation into the VT Concept & Requirements Exploration (C&RE) process. This analysis tool interfaces with the deactivation diagram to apply probable damage to the vessel and calculate survivability metrics for individual ship designs developed as part of the Multi-Objective Genetic Optimization (MOGO) ship design process.

2.6.1 Deactivation Diagrams in Network Theory

Deactivation diagrams improve upon the structural-representative adjacency list by providing a pre-constructed unidirectional system connection layout to support simultaneous evaluation of all connections in a multi-state system. Deactivation diagrams and adjacency list graphs are considerably similar for ship systems containing no bidirectional connections between components, but in other circumstances the two differ broadly. It is also important to note that the deactivation diagram explicitly identifies and maintains the connections within multi-path systems using logical (AND/OR) gates, whereas an adjacency list does not typically include this additional information.

Building upon system adjacency data, which often only include undirected edges providing no indication of directionality, the development of deactivation diagrams requires further analysis to include only directed connections that are utilized during anticipated ship operations. To do so, the Deactivation Diagram Tool pre-processes all data using the Depth-First Search (DFS) recursive algorithm to identify all valid directed system flow patterns and reduces the adjacency list to represent only valid paths before passing the source data along to the unidirectional representation algorithm.

2.6.2 Tool Development

In developing a deactivation diagram layout, existing system components are restructured and the introduction of new components and paths is often required to complete the necessary changes to the system architecture so that all applicable dependencies are retained. To do so, vertices that connect multi-directional paths are abstracted from the system diagram and inserted as VCs for a derived series of substitute unidirectional flow systems. An analysis class in the tool handles looping through each vertex and applies changes to the network structure based on the number of parents, children, flow patterns, and other parameters according a predefined algorithm.

Certain requirements have been set in place for the analysis of systems in the VT C&RE process and are adhered to in the development of deactivation diagrams for the system. The most prevalent of these is the distinction between VCs and SYS's within the layout. Each of these has specific behaviors in the deactivation diagram analysis and must be updated to fit the set of rules governing the analysis. For more information on the distinctions between VCs and SYS's in the VT deactivation diagram, see [9].

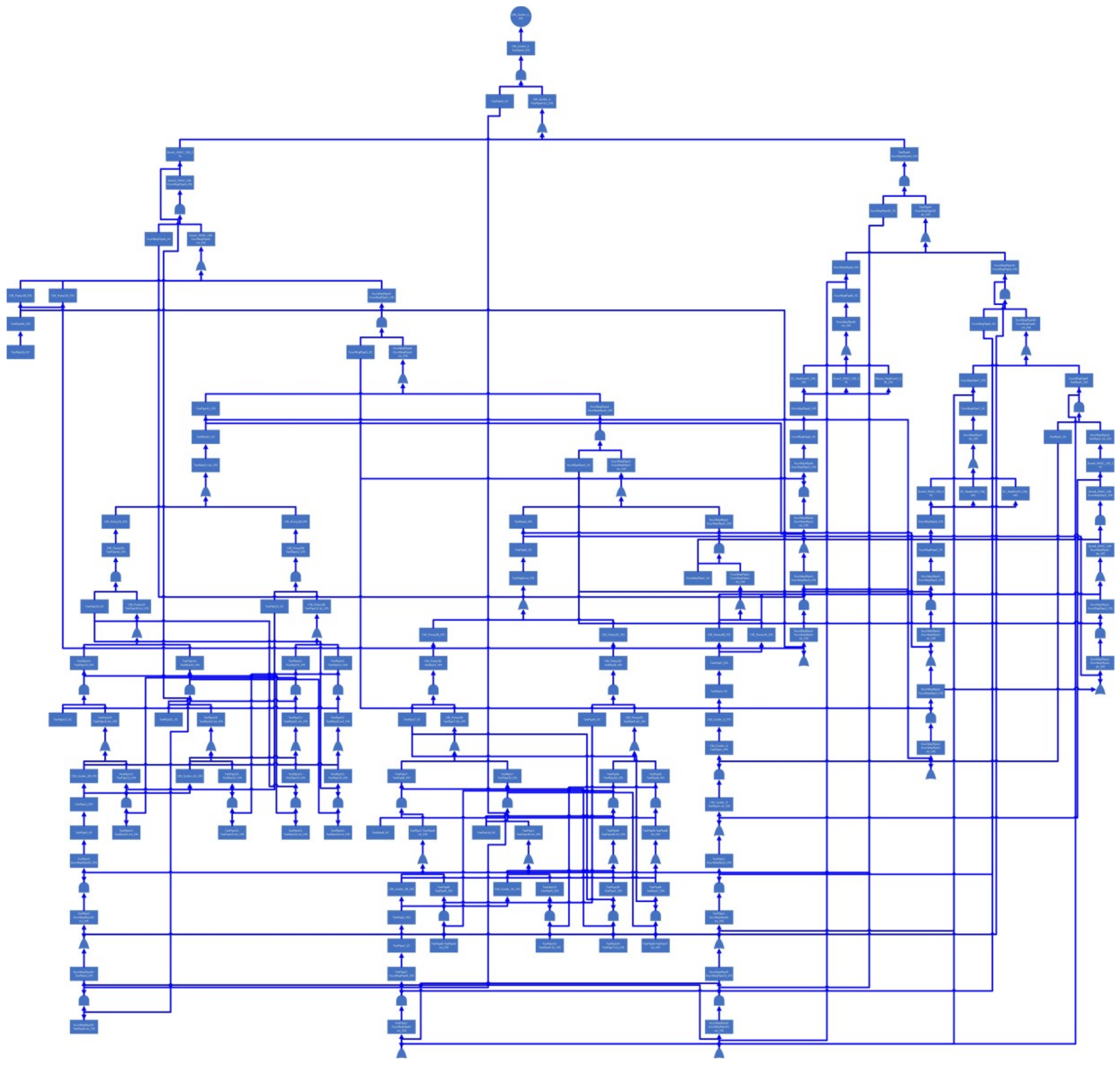


Figure 2.9: Visio Single-Sink Chilled Water Deactivation Diagram

Consistent with the current design of VT SSM tools, the Deactivation Diagram Tool was developed using Microsoft Visual Basic for Applications (VBA) embedded in Excel. This tool was developed for incorporation into the VT C&RE process as a supporting tool for preparing ship system data for further analysis. Input data provided to the tool exists in structured worksheet data or the Pajek ".net" network file format. Results of this tool are saved within in the containing workbook and are optionally exported back to a ".net" file and/or to representative diagrams in Microsoft Visio (where supported). Figure 2.9 shows an example of the Visio deactivation diagram output for the AFO Chilled Water System.

Figure 2.10 presents the results of the complete deactivation diagram analysis of one variant of the mechanical propulsion system developed by VT, previously shown in Figure 2.3. The Propulsion_SYS capability node, at the top of the figure, is followed by the totality of the ship system multiplex directly supporting the MECH plex. Large ship systems containing multiple plexes like the one shown present a significant challenge for the validation of the tool operation.

Despite manual verification of the Deactivation Diagram Tool utilizing small network diagram samples to ensure expected behavior, validation of large systems is limited by the scope of the analysis. Total system validation of complex systems currently relies heavily on system path checking for consistency and localized verification checks for diagram accuracy. Future efforts may include reverse-engineering analyses of deactivation diagrams into adjacency matrices for improved system comparison and validation.

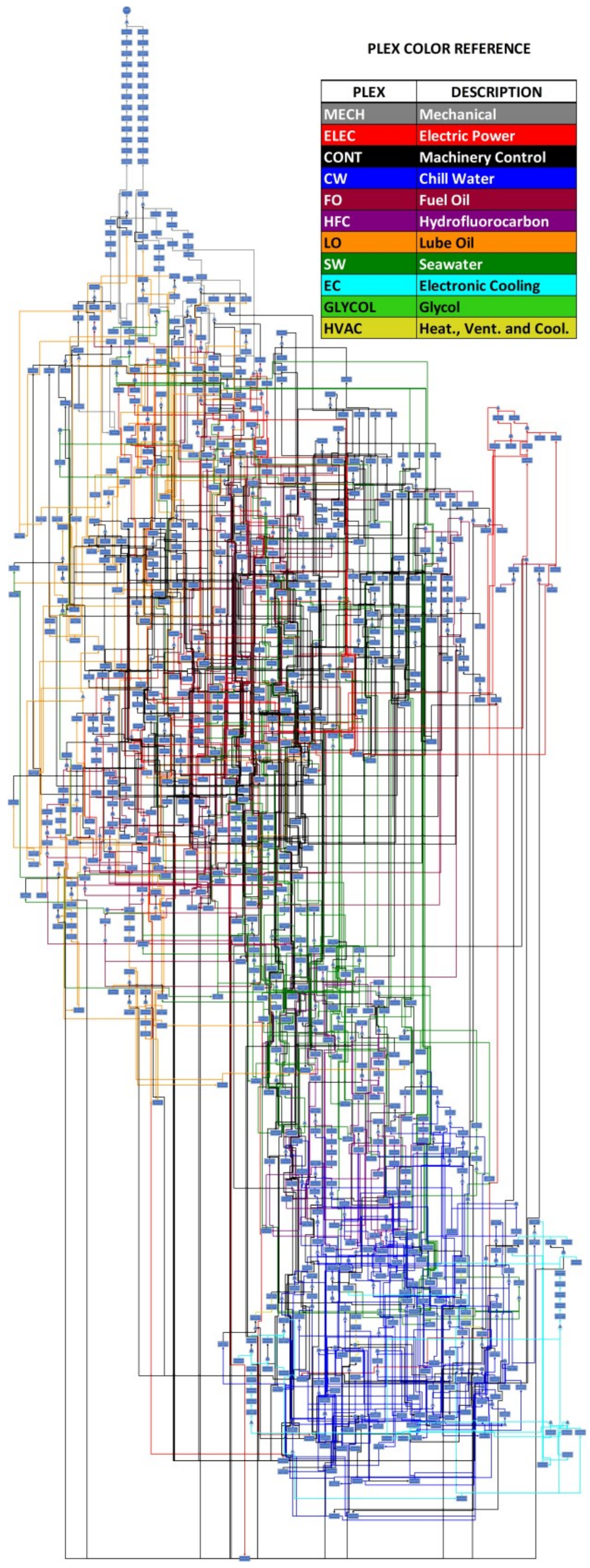
2.7 Conclusions

This paper has presented a methodology for extracting network definitions of systems stored in a FOCUS-compliant database and demonstrated how the architecture framework for distributed systems may be implemented in areas of system validation, equipment sizing, vulnerability, reliability, and network design. Figures included in this paper present an example workflow for a system described in LEAPS and utilized by external tools. Developments described in this paper were only possible through the combined efforts of multiple research groups and utilizing the consistent data structure of the LEAPS database.

Software tools presented in this paper have been developed to efficiently access and process the LEAPS database to promote network analysis. Several assumptions were made regarding the purpose of ports/terminals based on observed characteristics, leading to hard-coded object references for system directionality and component roles (sources/sinks). The current FOCUS PMM lacks a standardized methodology for identifying the directionality through ports/terminals; adding standardized port flow direction would greatly increase the flexibility of derived analysis tools to handle unidentified and future components.

Further, the concept of operational system nodes, which capture system operational requirements tied to a logical structure but not necessarily to a physical component or location, is not currently available in LEAPS. The ontology for storage of such a concept in LEAPS needs to be explored and defined.

To date, the development of network analysis tools has remained reliant on predetermined system characteristics and interoperability through external data files. We have shown that most of the source data necessary to run these programs is available in LEAPS and has the potential to be made more available to the end user. With the recommended changes to identify directionality and operational nodes in the FOCUS PMM, the door will be opened to future incorporation of existing network analysis tools into an integrated LEAPS-compatible environment.



PLEX	DESCRIPTION
MECH	Mechanical
ELEC	Electric Power
CONT	Machinery Control
CW	Chill Water
FO	Fuel Oil
HFC	Hydrofluorocarbon
LO	Lube Oil
SW	Seawater
EC	Electronic Cooling
GLYCOL	Glycol
HVAC	Heat., Vent. and Cool.

Figure 2.10: Mechanical Propulsion Operational System / Deactivation Diagram

2.8 References

[1] Naval Surface Warfare Center Design Tool Development Branch, *LEAPS Editor User's Manual Version 5.0*. West Bethesda, MD: Naval Surface Warfare Center Carderock Division, 2015. User's Manual available with LEAPS distribution.

[2] D. Brefort *et al.*, "An Architectural Framework For Distributed Naval Ship Systems," *Ocean Engineering,* vol. 147, pp. 375-385, 1 Jan. 2018.

[3] M. A. Parsons *et al.*, "Application of a Distributed System Architecture Framework to Naval Ship Concept and Requirements Exploration (C&RE)," in *Proc. ASNE Intelligent Ship Symposium*, 2019, in press.

[4] A. J. Brown, "Ship and Marine Engineering System Concept Design," in *Marine Engineering*, M. G. Parsons, Ed. 4th ed. Alexandria, VA: SNAME, in press, pp. 1-39.

[5] M. A. Parsons *et al.*, "Early-Stage Naval Ship Distributed System Design Using Architecture Flow Optimization," *Naval Engineers Journal*, unpublished.

[6] R. A. Dougal and B. Langland, "Catching it Early: Modeling and Simulating Distributed Systems in Early Stage Design," *Marine Technology,* pp. 63-69, Jan. 2016.

[7] J. Chalfant, Z. Wang, and M. Triantafyllou, "Expanding the Design Space Explored by S3D," in *Proc. 2019 IEEE Electric Ship Technologies Symposium (ESTS)*, in press.

[8] J. Chalfant, C. Chryssostomidis, D. J. Snyder, M. A. Parsons, and A. J. Brown, "Graph Theory Applications in FOCUS-Compliant Ship Designs," in *Proc. 2017 IEEE Electric Ship Technologies Symposium (ESTS)*, pp. 471-477.

[9] D. B. Goodfriend, "Exploration of System Vulnerability in Naval Ship Concept Design," M.S. Thesis, Department of Aerospace and Ocean Engineering, Virginia Polytechnic and State University, Blacksburg, VA, 2015.

Chapter 3

Naval Ship System Deactivation Analysis Using Network Architecture Framework

(Prepared for submission to the Journal of Complex Networks)

Daniel J. Snyder, Mustafa Y. Kara, Mark A. Parsons, and Alan J. Brown
Kevin T. Crofton Dept. of Aerospace and Ocean Engineering
Virginia Tech
Blacksburg, VA, USA

Abstract

Ship survivability has been identified by the United States Navy as a "fundamental design requirement" [1], and fleet reliability, maintainability, and availability (RMA) are always important issues with major impact on life cycle cost and readiness. Recent work completed on deactivation analysis at Virginia Tech (VT) supports a critical need to perform vulnerability and reliability analyses in early-stage ship design in the context of a new network architecture framework. In this work, an automated path-finding tool based on recursive data algorithms is presented for the analysis of ship systems and deactivation diagram building. Utilizing object-oriented programming, this innovative tool is robust and flexible for use in early-stage ship design with potential for future extension.

Keywords — *Deactivation Diagram, Network Theory, Ship System Design, Preliminary Design*

3.1 Introduction

Ongoing research exploring the use of networks in ship system design by the Naval International Cooperative Opportunities in Science and Technology program (NICOP) has paved the way for new efforts in addressing preliminary distributed system design and analysis in early-stage ship design using a new distributed system architecture framework. This framework, described in detail by Brefort et al [2], decomposes system architecture into three primary views: physical, logical, and operational. These views and their intersections provide the context for developing network-based tools to efficiently explore and analyze the ship system design space.

This work is supported by the Office of Naval Research (ONR) under ONR N00014-15-1-2476 (NICOP) Naval Ship Distributed System Vulnerability and Battle Damage Recovery in Early-Stage Ship Design and by the US Navy Naval Engineering Educational Consortium (NEEC).

Additional work on applications of this architecture framework is provided by Parsons et al. [3] in support of the Concept and Requirements Exploration (C&RE) process for ship design being developed at Virginia Tech (VT).

This paper builds on this new architecture framework by demonstrating novel ways to develop network views and analyzing connectivity and flow in distributed ship systems. An improved methodology for the analysis of Combat, Power, and Energy Systems (CPES) enables efficient exploration of large-scale ship system designs. Exploration of ship system architecture detailed in Section 3.2 enhances knowledge, encourages innovation, and supports the synthesis of affordable, effective ship designs.

The work detailed in Section 3.3 underlines the importance of system deactivation analyses and introduces a methodology for transforming a network architecture ship system description into a deactivation diagram for use in early-stage ship design evaluation. These efforts help to reduce the burden of manual creation of deactivation diagrams for complex ship systems, improving the efficiency of preliminary analyses and enabling design optimization algorithms. Custom analysis processes and criteria have been developed for optimal ship system processing in a non-integrated data environment, harnessing text-based system data and external input files.

The deactivation diagram analysis tool development described in Section 3.4 draws heavily on network theory to accurately develop and present the deactivation diagram for use in vulnerability and reliability analyses. This paper provides an abbreviated introduction to network theory and recursive algorithms as background for the analysis methodology and as clarification of assumptions and rules made regarding the input system data. In addition to primary development, this tool has been expanded for incorporation with LEAPS databases [4] and automated visualization. Examples of input files for the tool are detailed and results from the deactivation analysis are shown in tabular and graphical form in Section 3.6. Finally, both practical and theoretical uses for the tools described are presented and conclusions are drawn regarding the usefulness of the tool and future development efforts.

3.2 Architectural Framework

Brefort et al. [2] describes distributed system architecture as decomposable into physical, logical, and operational views as shown in Figure 3.1 [4]. According to Brefort et al.:

> This representation describes the spatial and functional relationships of the system together with their temporal behavior characteristics […]. The physical architecture describes the spatial arrangement, the logical architecture describes information on the functional characteristics of the system, and the operational architecture contains information on the temporal behavioral characteristics of the vessel, in a given mission scenario. [2, pp. 375-377]

This framework enables individual architectures to be implemented separately as illustrated in Figure 3.2 and integrated to solve for physical solutions, physical behavior, functional utilization and ultimately the total system response.

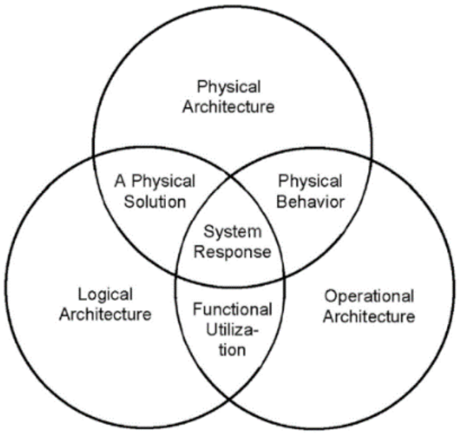


Figure 3.1: Representation of an Architectural Framework for Ship Distributed Systems [2]

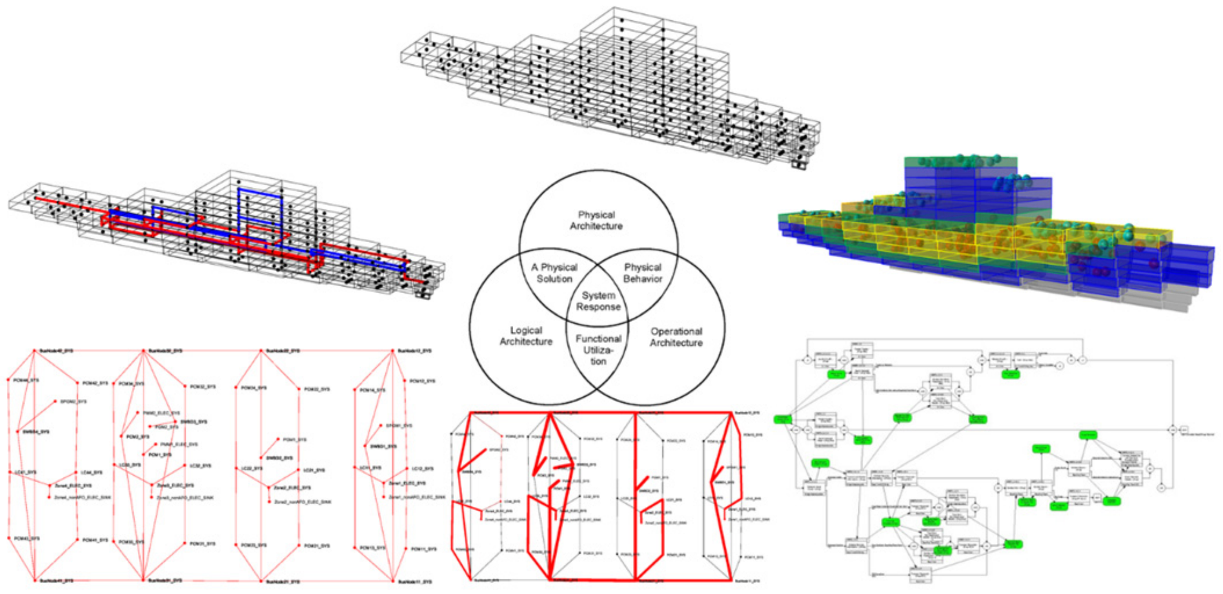


Figure 3.2: Notional Architecture Framework Implementation in Ship Concept Exploration [3]

The network logical architecture is the most fundamental component of this architecture framework. Figure 3.3 shows a simple logical system architecture of a mechanical subsystem or *plex* of a larger integrated power system. This network representation is made up of nodes and the connections between them, in which each node is either a vital component ("VC") or system node ("SYS"). A system node may represent a source, sink, or a vital component port in another plex. Each VC in this architecture also has physical attributes and is ultimately located physically in the ship in the "physical solution."

The Propulsion_SYS node shown in Figure 3.3 is an operational system node that provides propulsion capability to the operational architecture and pulls energy from the mechanical energy plex and ultimately the entire ship system multiplex to support a required level of performance, in this case meeting a designated ship propulsion speed. In order to operate, the mechanical subsystem shown in Figure 3.3 requires functional capability of other plexes within the multiplex system, including electric power, machinery control, lube oil, HVAC, seawater, chilled water, and HFC. When viewed in a multiplex deactivation diagram, these systems define the total ship propulsion system as shown in Figure 3.23.

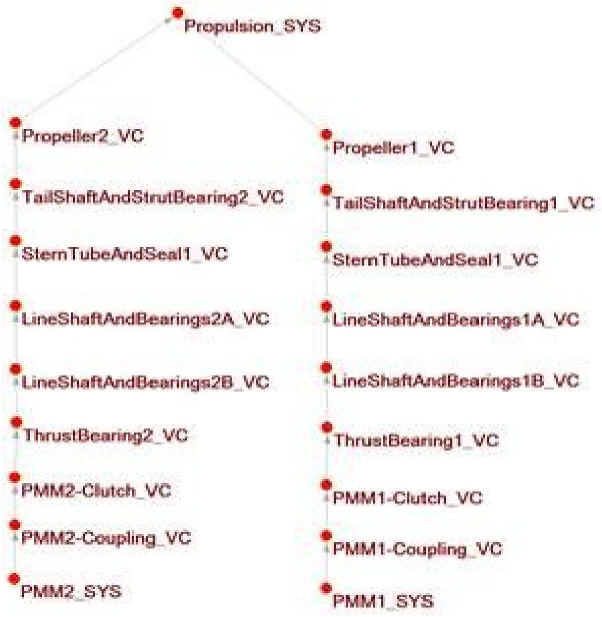


Figure 3.3: Mechanical Propulsion Plex Logical Architecture [4]

The network architecture framework facilitates useful applications such as Architecture Flow Optimization (AFO). An AFO begins with a system logical architecture, applies and enforces steady-state or quasi-steady-state operational constraints. The objective function in this optimization minimizes the flow cost of the network. This cost has two components: a fixed cost (representing the engineering and instillation costs of connecting components) and a variable cost (which linearly increases with flow). Parsons et al. [5] and Brown [6] provide a complete description of the linear optimization formulation and the operational constraints.

An AFO has many uses in early ship design including architecture optimization, component sizing and system feasibility assessment, but it requires significant preparation and computer processing time. A deactivation diagram view of the same logical architecture provides a much simplified and faster approach to initial vulnerability and reliability analysis. Obtaining this deactivation diagram view and developing its basic application is the primary focus of this paper.

3.3 Systems Analysis

Traditional systems with discrete source-sink pairs, such as fuel oil delivery between storage and consumer, often have unidirectional flow characteristics and are designed for only one flow direction as per the design limitations of integrated components. With the introduction of distributed electric control and power systems as well as a greater push for survivability through fully-redundant systems, these ship systems have evolved to support greater multi-directional flow characteristics based on system state. Multi-directional systems are designed for the potential of system flow redirection to keep the system largely operational after experiencing localized damage or during routine maintenance. This is particularly the case for electric power systems such as the shipboard zonal electric distribution systems (ZEDS), which commonly exhibit full redundancy for every electric bus and may have thousands of valid electric power flow paths between power producers (“sources”) and consumers (“sinks”).

When processing ship system connectivity in intact and damaged states, there is a significant computational overhead associated with querying and tracing between system components. Various methodologies have been introduced to speed up and expand this process, including work currently being performed on flow analysis by Parsons et al [5]; however, the process is still disproportionately limiting when performing statistical damage analysis in total ship design. To improve upon computational throughput in such analyses, the valid flow paths may be preprocessed and enumerated to simplify other analyses to a table look-up form. This precomputed system connectivity data is captured in the “deactivation diagram” data structure further described in this section.

Efforts in developing automated processes for system deactivation analysis such as those presented in this paper enable new views of the logical system architecture that has been developed from an architectural framework perspective. This work allows enables new types of analyses not easily handled previously by other processes and it supports incorporation of vulnerability exploration in the C&RE synthesis model.

3.3.1 Network Theory

The system description is comprised of nodes with interlinking connections called “Arcs” and “Edges” which form the traditional building blocks to the connectivity database. Nodes within the system network architecture represent either vital components (VCs) or systems (SYSs). Vital components, defined by Goodfriend as “ship equipment vital to ship system capability” [7], appear in network diagrams with limited connectivity with other components and present discrete points of failure as a part of larger systems. SYS’s in the diagram deliver the interconnectivity basis for the total system (plex) description, providing path nodes for distributed connection paths to intersect within the plex and may contain numerous connections to associated VCs or other SYS’s.

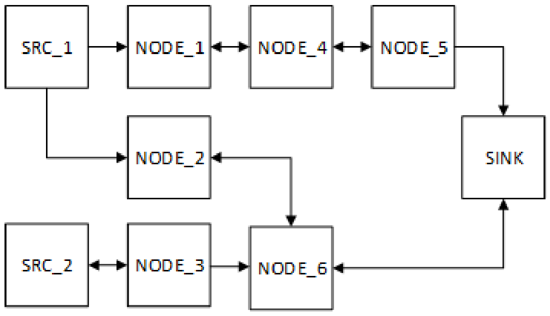
Arcs represent unidirectional connections between a parent (“source”) node and child (“sink”) node and edges represent bidirectional (undirected) connections between the two nodes. In the physical sense, an arc would be any mechanism connecting two components together that can only provide a “flow” in one direction, such as a one-way valve in a piping system. Some arcs may represent connections that are capable of providing feedback in the reverse direction but would not do so under any typical operating envelope, such as a shaft turning a generator. Edges represent all connections that have the design and/or capacity for bidirectional flow within the system and will perform accordingly in all anticipated system conditions. It is common in system analysis methodology to represent a bidirectional connection as a pair of opposing arcs in lieu of defining an edge for the sake of computational simplicity.

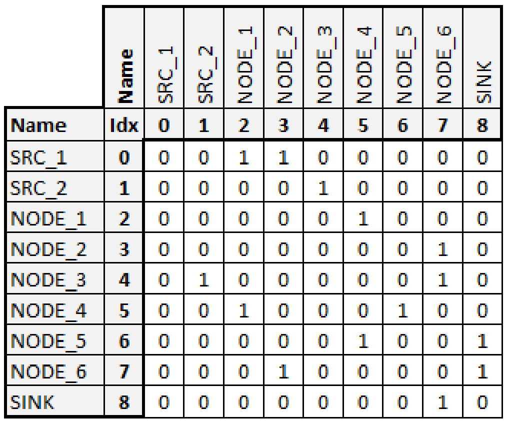
As a result of inconsistent terminology in network theory, other reference materials may sometimes use the term “Undirected Arc” instead of “Edge” to signify bidirectional connections. In these cases, unidirectional arcs are specifically referred to as “Directed Arcs”. In the context of this paper, the term “Arc” refers to an explicitly-defined unidirectional connection between a local source and sink.

3.3.2 Adjacency Matrices & Lists

The method described in this paper uses network data stored and exchanged in multiple types of system graphs. “Graphs” in this case refer to the mathematical data structures representing the connection relations within a list of known objects. These graphs contain data describing the directed connectivity between nodes in a simple format for data lookup and are useful when performing network analysis. The most common forms in this type of analysis are the adjacency matrix and adjacency list.

The adjacency matrix stores connection data for *n* nodes in a [n x n] square matrix with explicit definition of all connections between vertices. Vertices acting as sources to a connection are listed down the rows and the columns are sinks. Directionality is handled in an adjacency matrix by values placed on either side of the diagonal as a 1 (connected) or 0 (unconnected) as shown in Figure 3.4; however, non-binary numerals may be used to represent flow capacity or other metrics. An undirected system graph (bidirectional; edges only) is symmetric across the diagonal. This design provides beneficial memory-efficient data storage and simple index-based lookup for each connection data point.



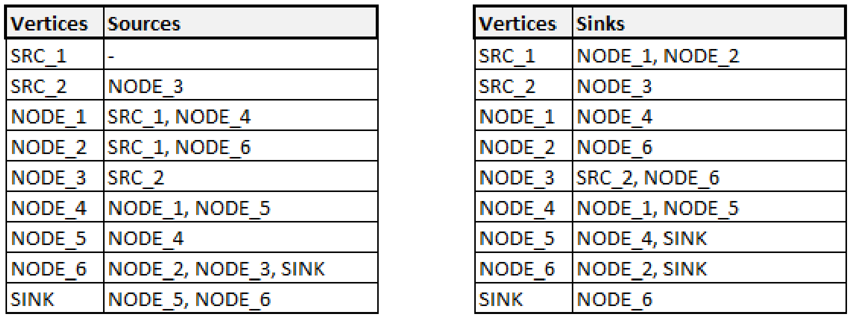


	Name	SRC_1	SRC_2	NODE_1	NODE_2	NODE_3	NODE_4	NODE_5	NODE_6	SINK
Name	**Idx**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
SRC_1	**0**	0	0	1	1	0	0	0	0	0
SRC_2	**1**	0	0	0	0	1	0	0	0	0
NODE_1	**2**	0	0	0	0	0	1	0	0	0
NODE_2	**3**	0	0	0	0	0	0	0	1	0
NODE_3	**4**	0	1	0	0	0	0	0	1	0
NODE_4	**5**	0	0	1	0	0	0	1	0	0
NODE_5	**6**	0	0	0	0	0	1	0	0	1
NODE_6	**7**	0	0	0	1	0	0	0	0	1
SINK	**8**	0	0	0	0	0	0	0	1	0

Figure 3.4: Example System Diagram and Adjacency Matrix

Within the ship system deactivation analysis framework, vertex ID numbers in the system diagram are commonly assigned using the Ship Work Breakdown Structure (SWBS) system and are not always sequential as a result. This results in a required secondary lookup table for matrix indices within the analysis and a hit to the runtime efficiency of the adjacency matrix due to extra work per computational cycle to extract the necessary data from the matrix.

An alternative to the adjacency matrix, the adjacency list stores the same information in a one-dimensional array of lists, each list containing all of the adjoining vertices for a specific vertex. Depending on the requirements of the analysis, this structure can list all sinks directly attached to a source or vice versa as shown in Figure 3.5. This adjacency list can be quickly looped over without checking every value in the matrix, thus greatly improving functional efficiency in network analysis. Drawbacks to the adjacency list include often increased memory overhead due to the less efficient manner of data storage and slower allocation speed for individual collection objects.



Vertices	Sources
SRC_1	-
SRC_2	NODE_3
NODE_1	SRC_1, NODE_4
NODE_2	SRC_1, NODE_6
NODE_3	SRC_2
NODE_4	NODE_1, NODE_5
NODE_5	NODE_4
NODE_6	NODE_2, NODE_3, SINK
SINK	NODE_5, NODE_6

Vertices	Sinks
SRC_1	NODE_1, NODE_2
SRC_2	NODE_3
NODE_1	NODE_4
NODE_2	NODE_6
NODE_3	SRC_2, NODE_6
NODE_4	NODE_1, NODE_5
NODE_5	NODE_4, SINK
NODE_6	NODE_2, SINK
SINK	NODE_6

Figure 3.5: Example Adjacency List Variants for System Shown in Figure 3.4

3.3.3 Vulnerability Analysis

This paper builds on the use of deactivation diagrams in vulnerability analysis performed by Goodfriend [7]. Although often deferred until after preliminary design, the comprehensive study of mission and overall vulnerability to damage from weapon effects is a major component of naval ship system design. It is well-established that changes to ship systems and structure during detail design incur significant additional costs, so there is merit in the inclusion of vulnerability analysis during or prior to preliminary design. This consideration of vulnerability in concept development also enables early evaluation of ship design robustness through the evaluation of damage propagation across ship systems and determination of subsequent system functionality.

Vulnerability analysis based on deactivation diagram input is a derivative of Success Tree Analysis, the logical converse of Fault Tree Analysis (FTA), intended to determine whether a given system will remain operational after damage is taken and one or more components of the system tree are rendered inoperable. The history of FTA dates back to its early use in the evaluation of the Minuteman Launch Control System (1961-62) by Hugh A. Watson of Bell Telephone Laboratories [8] and has since become ubiquitous in modern reliability and system analysis. This technique has been used across a wide variety of disciplines to identify risk factors and calculate cumulative risk assessments on given systems.

The Fault Tree Diagram (FTD) example in Figure 3.6 is developed based on a known system description and set of inputs to test the likelihood of an undesired outcome, such as the total failure of a system, and its connection with all foreseeable events that may result in such an outcome. This diagram describes the connections between unique events using Boolean “AND” and “OR” logic gates, indicating the combinations of events required to result in the undesired state. Using simple Boolean algebra, any combination of triggered events may be efficiently evaluated to determine if the undesired state is activated. In a deactivation diagram, these gates are used to describe the connections between adjacent nodes when multiple connection paths are converging at a common sink and represent the requirements for system operability based on vital connectivity.

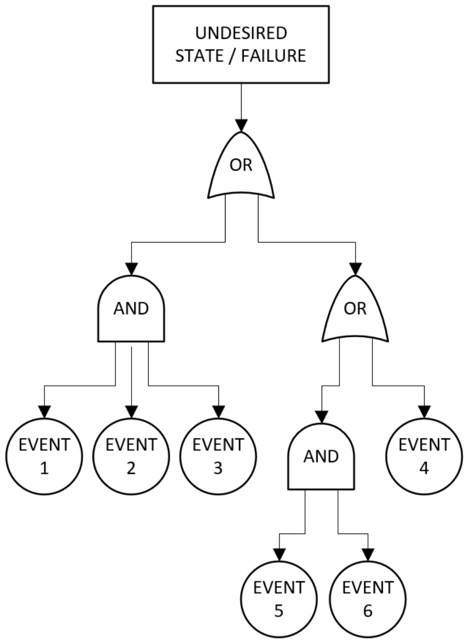


Figure 3.6: Fault Tree Diagram

Quantification of the vulnerability of large, distributed ship systems requires a meaningful statistical analysis undertaken over a wide spectrum of potential damage cases to appropriately capture the scope of damage effects [7]. At this time, large-scale statistical damage analyses face bottlenecks in the system analysis for case-by-case connectivity diagrams, which may be aided by the precomputation of deactivation diagrams. As a component of the overall measure of effectiveness (OMOE) in the Virginia Tech C&RE process, vulnerability plays a significant role in design space evaluation. Goodfriend and Brown [9] provide further discussion regarding system vulnerability in concept design.

3.3.4 Deactivation Diagrams

Until recently, ship system deactivation analysis in C&RE has primarily been a manual process and a chokepoint in the automation of system architectural analyses. Using deactivation processes, ship system analysis can develop adjacency relationships between components and collate network topology data into a practical form for further analysis and use in ship design evaluation. Results of this deactivation analysis form the deactivation diagram, a static categorization of all explicit connectivity paths within a system for determination of component operability based on the state of all vital source (parent) and intermediate components.

Deactivation diagrams, built on FTA and Reliability Block Diagram (RBD) theory, utilize the FTD design to provide a system description that may be rapidly evaluated to determine the viability of continued system operation. Figure 3.7 shows an RBD example layout of system components between the source object (in this case, the fuel storage tank) and the final consumer or “sink” (the MPE). Instead of being failure-state focused, as is the purpose of FTDs (otherwise known as negative analytical trees), the Deactivation Diagram identifies all sub-systems and related systems providing input to the target system. “Events” within the FTD are replaced by “nodes” (or “vertices”) within the deactivation diagram and any explicit flow direction between nodes is provided by arrow heads on the connections.

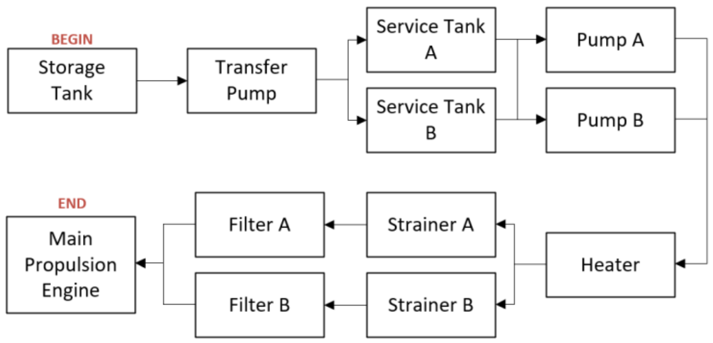


Figure 3.7: Example Fuel Oil RBD

Unidirectional flow systems, like the example system in Figure 3.7, require minimal processing to categorize and analyze connectivity. In this example, the fuel oil system described contains multiple parallel connection paths with predetermined flow directionality and only one of each parallel set is required for system operability, resulting in 27 (=3^3) potential flow paths (including sets of components operating in parallel). The RBD layout above is concise; however, the ambiguity of flow associated with the three and four-way connections does not support effective deactivation analysis.

The deactivation diagram removes ambiguity in system layouts by uniquely describing each connection member as a singular, unidirectional path. In systems with bidirectional flow, where flow between components (e.g. electric power, water, air, etc.) may move in either direction based on operating condition, the deactivation diagram contains discrete connections for each flow direction. The reduction in ambiguous connections results in a much more manageable system graph for rapid computational analyses at the expense of additional preprocessing and a minor data storage overhead. By including all valid paths in a deactivation diagram and enforcing one-way connectivity, the ship system deactivation analysis is also prevented from reaching an incalculable condition due to an invalid connection or infinite recursive data loop.

Each deactivation diagram is defined with a singular top node, the system sink, into which all valid flow paths converge. The sink node typically aligns with a specific ship function/capability (e.g. Propulsion, Fire Suppression, AAW, ASW, etc.). Maintaining the convergent nature of all event interactions within the FTD and developing discrete connections between the events and outcome is key to deactivation diagram development. Section 3.4 presents a methodology for developing new vertices in the deactivation diagram to ensure unique and easily-searchable flow patterns are provided.

Figure 3.8 shows a deactivation diagram example derived from the system network diagram shown in Figure 3.4. Unlike the system network, which allows for bidirectional connectivity between system nodes, the deactivation diagram is required to present a directed representation of the system network that only contains unidirectional arc connectivity between unique nodes. Modern ship systems with integrated redundant failure points have made it difficult to determine all valid paths allowable by the system description, so an advanced analysis methodology is required to generate an accurate system architecture that provides the deactivation diagram analysis with all viable system connection paths.

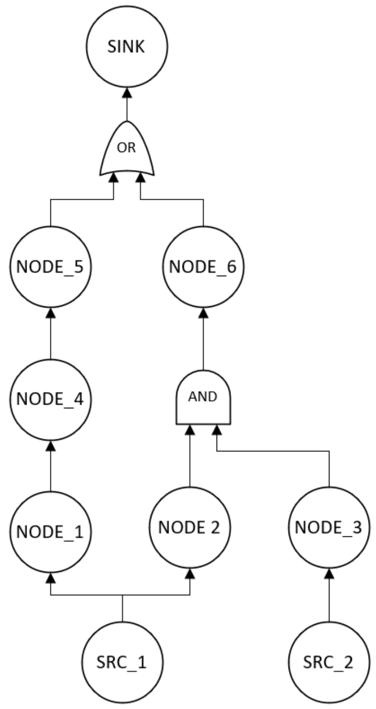


Figure 3.8: Example Deactivation Diagram

3.3.5 Graph Algorithms

The analysis of networks using graph exploration and decomposition has long been a topic of study in computer science. Elementary recursive methods, including Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms [10], are capable of thoroughly traversing entire graphs and more complex algorithms, including the Bellman–Ford Algorithm, Dijsktra's Algorithm, and Floyd-Warshall Algorithm [10], have been developed to more efficiently find the shortest paths between connected vertices.

To develop a deactivation diagram from an adjacency list/matrix, the problem must be approached with an effective search mechanism that records all valid connection paths between the predetermined source(s) and sink(s) and all paths must be recorded in a data object for future reference when developing the list of vertices of the finalized system. This methodology for system deactivation analysis precomputes all valid paths, requiring brute-force methodology that checks and tracks all paths to create a thorough exploration of detailed ship systems. The previously mentioned Dijsktra's Algorithm is widely known as an efficient method of finding the shortest path from sink to source in a distributed system analysis; however, its graph traversal technique does not lend itself towards the discovery and recording of all connected paths. For our purposes, the DFS and BFS algorithms are better suited to ensure full coverage of the system architecture.

The DFS algorithm takes the deep dive approach of network vertex discovery by searching along a single path as far as it can proceed until determining that the path is either valid or invalid. Once a potential connection path has reached its conclusion, the algorithm jumps back to the last "fork" or vertex with multiple system direction options. Any successful path discoveries are added directly onto a heap of paths by the recursive algorithm while the algorithm progresses to the next indexed node without any additional overhead-intensive tracking of where it has already searched. Alternatively, the BFS algorithm searches through nodes for connections based on total distance from the top node ("sink") and is an optimal approach for finding the shortest distance path from the top node. When used for flow analysis, this method holds a significant amount of data in memory while non-sequentially tracing each path and is less optimal for deactivation path analysis in memory-limited applications such as Microsoft Excel. This hierarchical graph traversal makes BFS better suited for incremental output of data, such as the visualization of network paths in Microsoft Visio described in Section 3.5. Figure 3.9 provides an example of the order of operations for both methods.

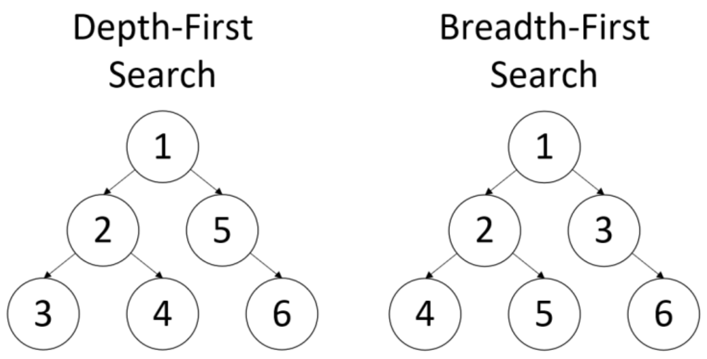


Figure 3.9: Algorithm Order of Search Operations

3.4 Analysis Tool

3.4.1 Data Analysis

The deactivation diagram approach presented in this paper supports a detailed ship deactivation analysis based on a global ship system model. To fully actualize all paths through bidirectional ship systems, such as a distributed electric power system shown in Figure 3.10, the system must be fully developed within an object-oriented analysis tool that can provide quick object lookup for processing vertex data. Due to concerns about the upkeep of the code by future teams and students, the decision was made early on to develop this deactivation diagram tool in Microsoft Visual Basic for Applications (VBA), an integral part of the Microsoft Office Suite. Specifically, this code would exist within VBA for Excel and would exchange input and output data with the Excel Workbook in which it is embedded.

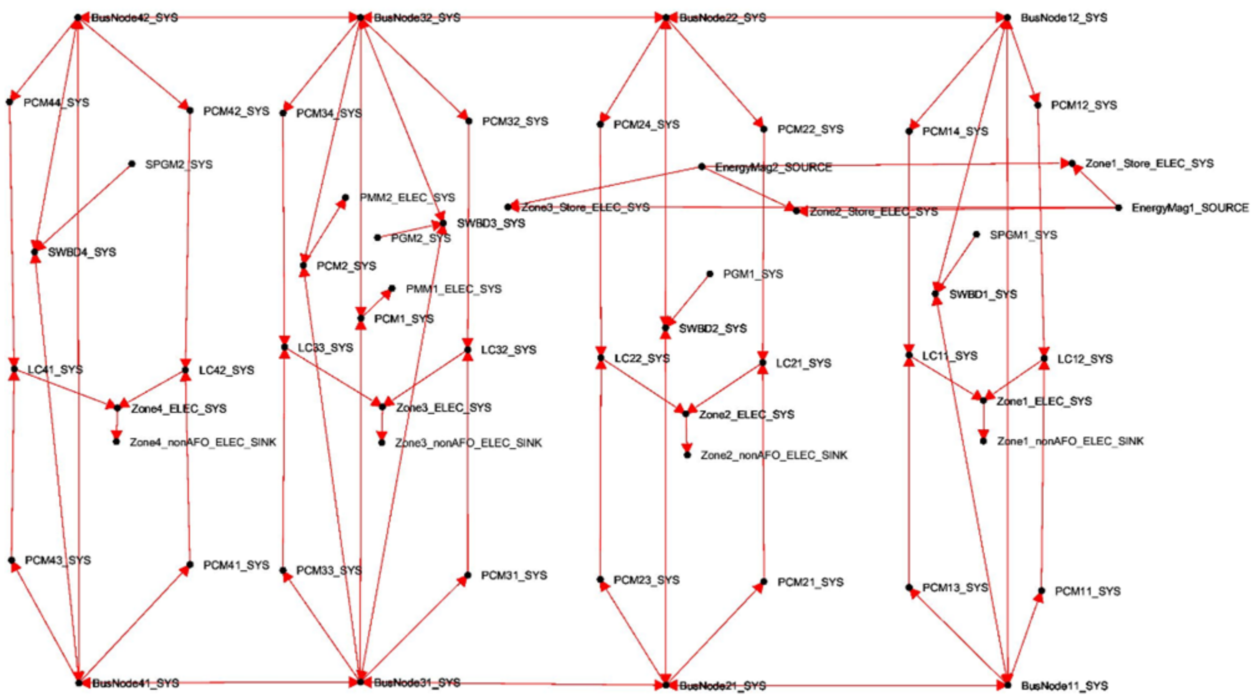


Figure 3.10: Zonal Electric Power Distribution Plex (ELEC) Diagram

Within the ship synthesis model (SSM) system data structure, each system vertex is associated with data pertaining to system attributes as well as its own connections to other vertices. Instead of creating and maintaining a database-type structure as a repository for data required in the analysis and by the final consumer, it was decided that each vertex should be developed in the computational model as an object that holds onto its own set of reference data and system requirements (*Data_Vertex_Obj*), containing the following information:

- Name
- ID Number
- Compartment
- Associated Plexes
- Source Logic Connection (AND/OR)
- Directionality
- Object Type (VC/SYS)
- Special Type (Source/Sink)

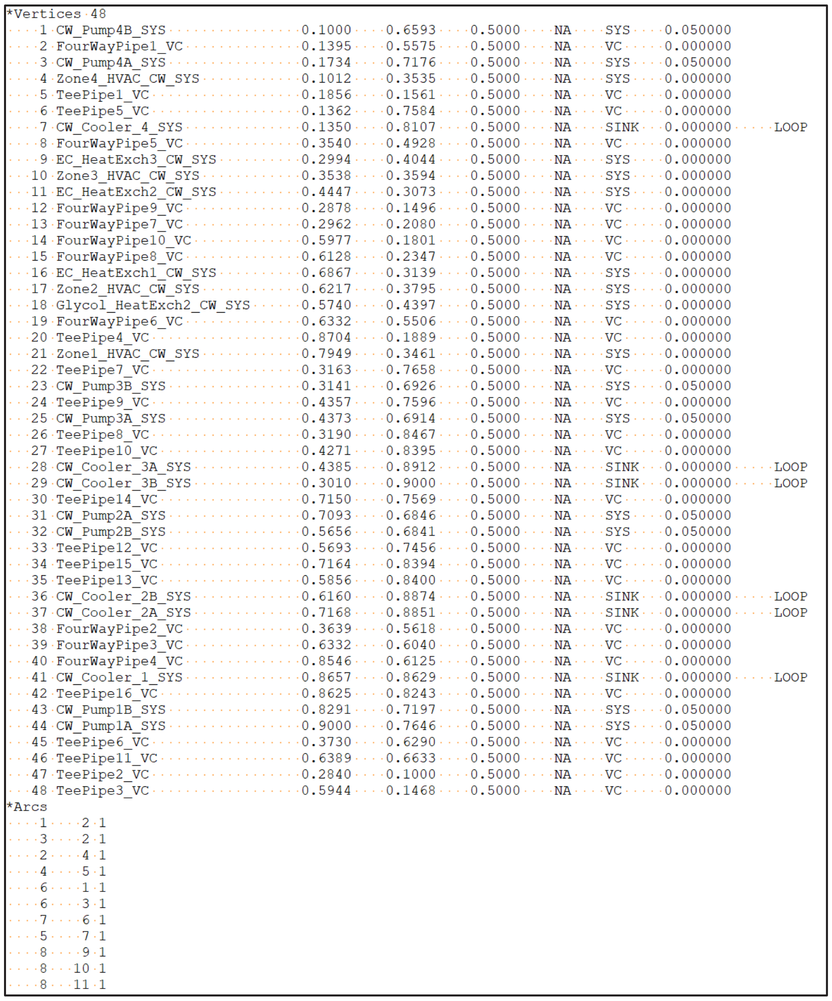
Using this object-oriented model reduces much of the record-keeping overhead in system analyses and provides extra robustness to the data structure by allowing for improved error handling and data consistency checking. This model was eventually developed into an organized encapsulation of system vertices, all directed connections, and system description information necessary to path-finding algorithms as a system data object (*Data_System_Obj*).

Data ingest to the deactivation analysis program from internal Excel data and external files occurs line-by-line and requires incremental caching of data until all information has been read into the model to finalize the complete data set. Additionally, most external data files are limited to definition of a single plex and some systems in the C&RE database exist on separate Excel worksheets, so data cumulation and post-process finalization is also a valid approach to dealing with issues that arise from this. The process used for this is to incrementally read each data source, create a unique system object for the source, and cache any vertices with missing attributes. These system objects are appended into a consolidated object and the data set is post-processed to fully populate all vertex information and finalize the complete system object. Any data errors during this process are identified and reported back to the user.

3.4.2 Input Parsing

Data input to the deactivation diagram tool was initially generated using the open-source Pajek ".net" network file format (e.g. "CW.net") and later expanded to incorporate additional data formats, including the VT SSM data structure in Excel. The ".net" file format example in Figure 3.11 contains five principal columns of organized data specific to the vertex definition required by Pajek: vertex number, name, and 3-D rectangular coordinates (x, y, z). This format was chosen for its interoperability with common network analysis tools (Pajek, Gephi, NodeXL) as well as file structure expandability to include non-restricted data stored beyond the five fixed input data columns as shown below. In Figure 3.11, secondary data specific to ship system connectivity and deactivation diagram development has been added to the right of the rectangular coordinate columns and include compartment, vertex type ("VC", "SYS", or "SINK"), power/flow requirements (not currently utilized), and system operation type (open/closed loop). Sinks that are part of a closed loop system are simply marked "LOOP" and open loop systems contain the source list attributed to the given sink vertex. Sources in the source list are typically not directly connected with the sink vertex and the paths between the source(s) and sink must be discovered through analysis.

Certain restrictions in data handling exist in current ".net" file formatting that must be considered. As previously mentioned, the current ".net" format allows for only one plex in each file and it does not offer a place-holder for the network name within the file, so the filename is substituted for the plex name in the deactivation analysis. Additionally, Pajek requires sequential vertex numbering starting at one which does not fit with SWBS-based numbering schemas. As a result, overlapping of vertex numbers is inevitable when multiple ".net" files are utilized in the deactivation analysis. This behavior is handled by careful redefinition of vertex numbers, arcs, and edges in the system data structure during post-processing of the system data object.

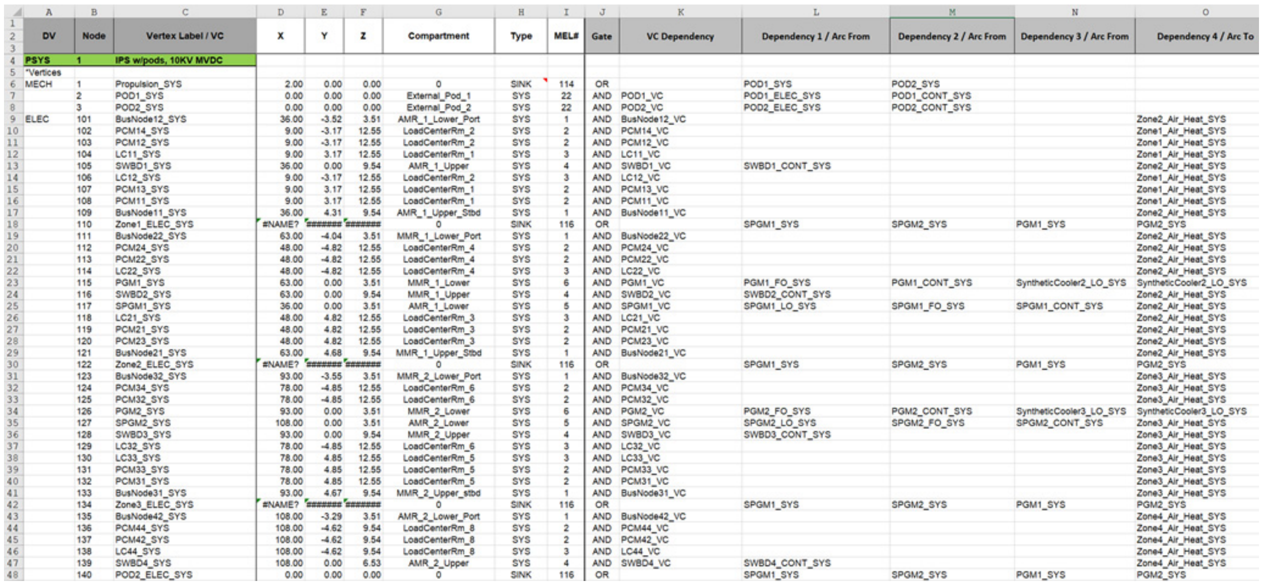


```
*Vertices 48
    1 CW_Pump4B_SYS                 0.1000    0.6593    0.5000    NA    SYS    0.050000
    2 FourWayPipe1_VC               0.1395    0.5575    0.5000    NA    VC     0.000000
    3 CW_Pump4A_SYS                 0.1734    0.7176    0.5000    NA    SYS    0.050000
    4 Zone4_HVAC_CW_SYS             0.1012    0.3535    0.5000    NA    SYS    0.000000
    5 TeePipe1_VC                   0.1856    0.1561    0.5000    NA    VC     0.000000
    6 TeePipe5_VC                   0.1362    0.7584    0.5000    NA    VC     0.000000
    7 CW_Cooler_4_SYS               0.1350    0.8107    0.5000    NA    SINK   0.000000     LOOP
    8 FourWayPipe5_VC               0.3540    0.4928    0.5000    NA    VC     0.000000
    9 EC_HeatExch3_CW_SYS           0.2994    0.4044    0.5000    NA    SYS    0.000000
   10 Zone3_HVAC_CW_SYS             0.3538    0.3594    0.5000    NA    SYS    0.000000
   11 EC_HeatExch2_CW_SYS           0.4447    0.3073    0.5000    NA    SYS    0.000000
   12 FourWayPipe9_VC               0.2878    0.1496    0.5000    NA    VC     0.000000
   13 FourWayPipe7_VC               0.2962    0.2080    0.5000    NA    VC     0.000000
   14 FourWayPipe10_VC              0.5977    0.1801    0.5000    NA    VC     0.000000
   15 FourWayPipe8_VC               0.6128    0.2347    0.5000    NA    VC     0.000000
   16 EC_HeatExch1_CW_SYS           0.6867    0.3139    0.5000    NA    SYS    0.000000
   17 Zone2_HVAC_CW_SYS             0.6217    0.3795    0.5000    NA    SYS    0.000000
   18 Glycol_HeatExch2_CW_SYS       0.5740    0.4397    0.5000    NA    SYS    0.000000
   19 FourWayPipe6_VC               0.6332    0.5506    0.5000    NA    VC     0.000000
   20 TeePipe4_VC                   0.8704    0.1889    0.5000    NA    VC     0.000000
   21 Zone1_HVAC_CW_SYS             0.7949    0.3461    0.5000    NA    SYS    0.000000
   22 TeePipe7_VC                   0.3163    0.7658    0.5000    NA    VC     0.000000
   23 CW_Pump3B_SYS                 0.3141    0.6926    0.5000    NA    SYS    0.050000
   24 TeePipe9_VC                   0.4357    0.7596    0.5000    NA    VC     0.000000
   25 CW_Pump3A_SYS                 0.4373    0.6914    0.5000    NA    SYS    0.050000
   26 TeePipe8_VC                   0.3190    0.8467    0.5000    NA    VC     0.000000
   27 TeePipe10_VC                  0.4271    0.8395    0.5000    NA    VC     0.000000
   28 CW_Cooler_3A_SYS              0.4385    0.8912    0.5000    NA    SINK   0.000000     LOOP
   29 CW_Cooler_3B_SYS              0.3010    0.9000    0.5000    NA    SINK   0.000000     LOOP
   30 TeePipe14_VC                  0.7150    0.7569    0.5000    NA    VC     0.000000
   31 CW_Pump2A_SYS                 0.7093    0.6846    0.5000    NA    SYS    0.050000
   32 CW_Pump2B_SYS                 0.5656    0.6841    0.5000    NA    SYS    0.050000
   33 TeePipe12_VC                  0.5693    0.7456    0.5000    NA    VC     0.000000
   34 TeePipe15_VC                  0.7164    0.8394    0.5000    NA    VC     0.000000
   35 TeePipe13_VC                  0.5856    0.8400    0.5000    NA    VC     0.000000
   36 CW_Cooler_2B_SYS              0.6160    0.8874    0.5000    NA    SINK   0.000000     LOOP
   37 CW_Cooler_2A_SYS              0.7168    0.8851    0.5000    NA    SINK   0.000000     LOOP
   38 FourWayPipe2_VC               0.3639    0.5618    0.5000    NA    VC     0.000000
   39 FourWayPipe3_VC               0.6332    0.6040    0.5000    NA    VC     0.000000
   40 FourWayPipe4_VC               0.8546    0.6125    0.5000    NA    VC     0.000000
   41 CW_Cooler_1_SYS               0.8657    0.8629    0.5000    NA    SINK   0.000000     LOOP
   42 TeePipe16_VC                  0.8625    0.8243    0.5000    NA    VC     0.000000
   43 CW_Pump1B_SYS                 0.8291    0.7197    0.5000    NA    SYS    0.050000
   44 CW_Pump1A_SYS                 0.9000    0.7646    0.5000    NA    SYS    0.050000
   45 TeePipe6_VC                   0.3730    0.6290    0.5000    NA    VC     0.000000
   46 TeePipe11_VC                  0.6389    0.6633    0.5000    NA    VC     0.000000
   47 TeePipe2_VC                   0.2840    0.1000    0.5000    NA    VC     0.000000
   48 TeePipe3_VC                   0.5944    0.1468    0.5000    NA    VC     0.000000
*Arcs
    1    2 1
    3    2 1
    2    4 1
    4    5 1
    6    1 1
    6    3 1
    7    6 1
    5    7 1
    8    9 1
    8   10 1
    8   11 1
```

Figure 3.11: Sample Chilled Water System (CW) Data Structure in Pajek

In lieu of data exchange through external files, support for the VT SSM resulted in the development of a fixed internal Excel format for passing data into the deactivation analysis. This alternative format was developed to closely mimic the ".net" file format, as shown in Figure 3.12. Within the Excel worksheet, the vertices are individually listed along with specific information regarding object type, location, role, and dependencies. In this case, dependencies shown to the right of each vertex include direct (explicit) vital component dependencies, implicit dependencies to other plexes, and (for sink vertices) plex source dependencies. Just like the ".net" file format, the list of vertex names is followed by a list of explicit directed vertex connections.

The predetermined pattern of assigning connectivity within a plex, hereby referred to as "explicit" connectivity, is to list all nodal connections in a list following the node definitions. This structure is well-organized and well-suited for object-oriented code development; however, it requires additional data structures to handle processing of the connections separate from the vertex definitions.



	DV	Node	Vertex Label / VC	X	Y	Z	Compartment	Type	MEL#	Gate	VC Dependency	Dependency 1 / Arc From	Dependency 2 / Arc From	Dependency 3 / Arc From	Dependency 4 / Arc To
4	PSYS	1	IPS w/pods, 10KV MVDC												
5	*Vertices														
6	MECH	1	Propulsion_SYS	2.00	0.00	0.00	0	SINK	114	OR		POD1_SYS	POD2_SYS		
7		2	POD1_SYS	0.00	0.00	0.00	External_Pod_1	SYS	22	AND	POD1_VC	POD1_ELEC_SYS	POD1_CONT_SYS		
8		3	POD2_SYS	0.00	0.00	0.00	External_Pod_2	SYS	22	AND	POD2_VC	POD2_ELEC_SYS	POD2_CONT_SYS		
9	ELEC	101	BusNode12_SYS	36.00	-3.52	3.51	AMR_1_Lower_Port	SYS	1	AND	BusNode12_VC				Zone2_Air_Heat_SYS
10		102	PCM14_SYS	9.00	-3.17	12.55	LoadCenterRm_2	SYS	2	AND	PCM14_VC				Zone1_Air_Heat_SYS
11		103	PCM12_SYS	9.00	-3.17	12.55	LoadCenterRm_2	SYS	2	AND	PCM12_VC				Zone1_Air_Heat_SYS
12		104	LC11_SYS	9.00	3.17	12.55	LoadCenterRm_1	SYS	3	AND	LC11_VC				Zone1_Air_Heat_SYS
13		105	SWBD1_SYS	36.00	0.00	9.54	AMR_1_Upper	SYS	4	AND	SWBD1_VC	SWBD1_CONT_SYS			Zone2_Air_Heat_SYS
14		106	LC12_SYS	9.00	-3.17	12.55	LoadCenterRm_2	SYS	3	AND	LC12_VC				Zone1_Air_Heat_SYS
15		107	PCM13_SYS	9.00	3.17	12.55	LoadCenterRm_1	SYS	2	AND	PCM13_VC				Zone1_Air_Heat_SYS
16		108	PCM11_SYS	9.00	3.17	12.55	LoadCenterRm_1	SYS	2	AND	PCM11_VC				Zone1_Air_Heat_SYS
17		109	BusNode11_SYS	36.00	4.31	9.54	AMR_1_Upper_Stbd	SYS	1	AND	BusNode11_VC				Zone2_Air_Heat_SYS
18		110	Zone1_ELEC_SYS	#NAME?	#######	#######	0	SINK	116	OR		SPGM1_SYS	SPGM2_SYS	PGM1_SYS	PGM2_SYS
19		111	BusNode22_SYS	63.00	-4.04	3.51	MMR_1_Lower_Port	SYS	1	AND	BusNode22_VC				Zone2_Air_Heat_SYS
20		112	PCM24_SYS	48.00	-4.82	12.55	LoadCenterRm_4	SYS	2	AND	PCM24_VC				Zone2_Air_Heat_SYS
21		113	PCM22_SYS	48.00	-4.82	12.55	LoadCenterRm_4	SYS	2	AND	PCM22_VC				Zone2_Air_Heat_SYS
22		114	LC22_SYS	48.00	-4.82	12.55	LoadCenterRm_4	SYS	3	AND	LC22_VC				Zone2_Air_Heat_SYS
23		115	PGM1_SYS	63.00	0.00	3.51	MMR_1_Lower	SYS	6	AND	PGM1_VC	PGM1_FO_SYS	PGM1_CONT_SYS	SyntheticCooler2_LO_SYS	SyntheticCooler2_LO_SYS
24		116	SWBD2_SYS	63.00	0.00	9.54	MMR_1_Upper	SYS	4	AND	SWBD2_VC	SWBD2_CONT_SYS			Zone2_Air_Heat_SYS
25		117	SPGM1_SYS	36.00	0.00	3.51	AMR_1_Lower	SYS	5	AND	SPGM1_VC	SPGM1_LO_SYS	SPGM1_FO_SYS	SPGM1_CONT_SYS	Zone2_Air_Heat_SYS
26		118	LC21_SYS	48.00	4.82	12.55	LoadCenterRm_3	SYS	3	AND	LC21_VC				Zone2_Air_Heat_SYS
27		119	PCM21_SYS	48.00	4.82	12.55	LoadCenterRm_3	SYS	2	AND	PCM21_VC				Zone2_Air_Heat_SYS
28		120	PCM23_SYS	48.00	4.82	12.55	LoadCenterRm_3	SYS	2	AND	PCM23_VC				Zone2_Air_Heat_SYS
29		121	BusNode21_SYS	63.00	4.68	9.54	MMR_1_Upper_Stbd	SYS	1	AND	BusNode21_VC				Zone2_Air_Heat_SYS
30		122	Zone2_ELEC_SYS	#NAME?	#######	#######	0	SINK	116	OR		SPGM1_SYS	SPGM2_SYS	PGM1_SYS	PGM2_SYS
31		123	BusNode32_SYS	93.00	-3.55	3.51	MMR_2_Lower_Port	SYS	1	AND	BusNode32_VC				Zone3_Air_Heat_SYS
32		124	PCM34_SYS	78.00	-4.85	12.55	LoadCenterRm_6	SYS	2	AND	PCM34_VC				Zone3_Air_Heat_SYS
33		125	PCM32_SYS	78.00	-4.85	12.55	LoadCenterRm_6	SYS	2	AND	PCM32_VC				Zone3_Air_Heat_SYS
34		126	PGM2_SYS	93.00	0.00	3.51	MMR_2_Lower	SYS	6	AND	PGM2_VC	PGM2_FO_SYS	PGM2_CONT_SYS	SyntheticCooler3_LO_SYS	SyntheticCooler3_LO_SYS
35		127	SPGM2_SYS	108.00	0.00	3.51	AMR_2_Lower	SYS	5	AND	SPGM2_VC	SPGM2_LO_SYS	SPGM2_FO_SYS	SPGM2_CONT_SYS	Zone3_Air_Heat_SYS
36		128	SWBD3_SYS	93.00	0.00	9.54	MMR_2_Upper	SYS	4	AND	SWBD3_VC	SWBD3_CONT_SYS			Zone3_Air_Heat_SYS
37		129	LC32_SYS	78.00	-4.85	12.55	LoadCenterRm_6	SYS	3	AND	LC32_VC				Zone3_Air_Heat_SYS
38		130	LC33_SYS	78.00	4.85	12.55	LoadCenterRm_5	SYS	3	AND	LC33_VC				Zone3_Air_Heat_SYS
39		131	PCM33_SYS	78.00	4.85	12.55	LoadCenterRm_5	SYS	2	AND	PCM33_VC				Zone3_Air_Heat_SYS
40		132	PCM31_SYS	78.00	4.85	12.55	LoadCenterRm_5	SYS	2	AND	PCM31_VC				Zone3_Air_Heat_SYS
41		133	BusNode31_SYS	93.00	4.67	9.54	MMR_2_Upper_stbd	SYS	1	AND	BusNode31_VC				Zone3_Air_Heat_SYS
42		134	Zone3_ELEC_SYS	#NAME?	#######	#######	0	SINK	116	OR		SPGM1_SYS	SPGM2_SYS	PGM1_SYS	PGM2_SYS
43		135	BusNode42_SYS	108.00	-3.29	3.51	AMR_2_Lower_Port	SYS	1	AND	BusNode42_VC				Zone4_Air_Heat_SYS
44		136	PCM44_SYS	108.00	-4.62	9.54	LoadCenterRm_8	SYS	2	AND	PCM44_VC				Zone4_Air_Heat_SYS
45		137	PCM42_SYS	108.00	-4.62	9.54	LoadCenterRm_8	SYS	2	AND	PCM42_VC				Zone4_Air_Heat_SYS
46		138	LC44_SYS	108.00	-4.62	9.54	LoadCenterRm_8	SYS	3	AND	LC44_VC				Zone4_Air_Heat_SYS
47		139	SWBD4_SYS	108.00	0.00	6.53	AMR_2_Upper	SYS	4	AND	SWBD4_VC	SWBD4_CONT_SYS			Zone4_Air_Heat_SYS
48		140	POD2_ELEC_SYS	0.00	0.00	0.00	0	SINK	116	OR		SPGM1_SYS	SPGM2_SYS	PGM1_SYS	PGM2_SYS

Figure 3.12: Sample Propulsion System (PSYS) Data Structure in Excel

3.4.3 Path Finding

The path-finding algorithm requires appropriate bounding of the system flow to start the analysis and identify valid flow paths. When considering system flow paths for deactivation, the system operation must be initially identified as an open-loop or closed-loop system. Open-loop systems have distinct system sinks and sources framing the boundaries of the system while closed-loop systems identify one or more vertices as a dual sink/source for the roundtrip flow. Implicit assumptions regarding sink and source assignments often result in erroneous data, so these system sinks and sources are required to be explicitly identified in the input data.

The design of a path-finding algorithm seeks to balance robustness with computational speed and accuracy. Careful consideration of computational analysis methods led to the selection of the depth-first search (DFS) algorithm as the most appropriate method for path-finding operations within the deactivation analysis tool. This methodology yields predictable, linear data set analysis and maintains a low memory overhead when properly programmed for sequential data access. Advanced analysis methods, such as Dijsktra's Algorithm, are optimized for individual paths and lack the data scope necessary for total system tracking. In this tool, paths discovered using DFS are stored as arrays within a collection data object, easily referenceable from outside data sources and stored within the global deactivation data model.

The primary causes of failure in a recursion algorithm are invalid data references, memory overutilization, and infinite recursive data loops. With these issues in mind, this DFS-based path-finding algorithm was developed to be sufficiently simple and robust. The process does not consider Boolean gates in the connectivity patterns or flow rate and capacity through vertices, these details are outside the scope of the strict path-only finding analysis.

Before the DFS algorithm is processed in the path finding code, a thorough data pre-processor double-checks the directionality of existing connections and the quality of the data being analyzed to prevent data lookup failure and eliminate recursion loops. Memory issues are managed by the incremental storage of completed, valid deactivation paths in a dedicated collection instead of storing all paths including incomplete and invalid paths during runtime as would occur during a BFS path-finding algorithm. Recursion is prevented by marking all previously “visited” vertices within the scope of paths originating from a common sink. To prevent unexpected data errors leading to infinite recursion, a loop counter was additionally implemented to limit the number of cycles performed by the code and the escape key (“Esc”) was programmed to abort current analysis operations.

DFS analysis recursion is started from each sink rather than at a source, which has been shown through testing to be more reliable in the analysis and remains consistent with the deactivation approach to determining the overall effects on behavior at each plex sink. System data is precompiled into a reverse adjacency list containing all sources associated with each sink and fed into the DFS algorithm to quickly search based on the sequential index of each vertex in the total collection. The process loops recursively over each path node, querying the adjacency list for the subsequent vertices in the current path and continuing to loop over any vertices containing valid and unvisited sources. For closed-loop systems, the process successfully completes a single path if the starting vertex matches the ending vertex with more than two vertices in the system path. For open-loop systems, the process searches for preallotted sources based on the system description and stores the current path once any valid source has been reached. Dead ends in the analysis result in invalid paths and are not retained. With each path completed, the analysis drops back as far as necessary to find vertices along paths not yet visited by the code and continues recursion analysis until all valid flow paths have been developed as shown in Figure 3.13.

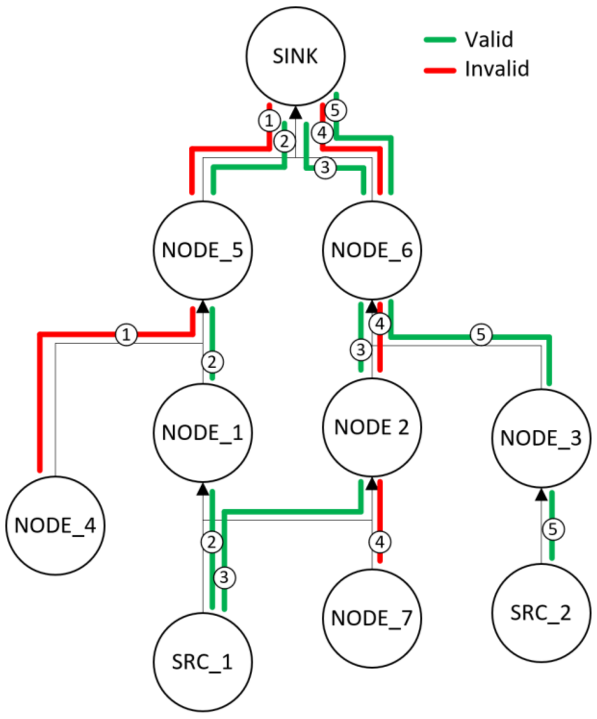


Figure 3.13: Order of DFS Path Search Operations from Sink To Sources

3.4.4 System Manipulation

The most significant part of this deactivation diagram development is the separation of bidirectional flow into unidirectional flow connections. This forms the basis of the qualified flow paths within the deactivation diagram, which requires that all connections must represent unidirectional flow between components. System manipulation methods in this tool serve the purpose of isolating unidirectional paths and developing vertices that can be uniquely identified within the system definition. The resultant system structure is compatible with general ship system deactivation analysis and the VT CR&E process

Vertices in network analysis are recorded using unique id numbers and names, which are used for data lookup while performing path and deactivation analysis. A non-unique vertex does not have demonstrate consistent flow properties for every operability case and may appear in multiple locations within the logical system architecture. Any unmodified vertices containing inconsistent flow connections within the network will result in invalid data operations and cause list-based system connectivity analysis to fail. The most common cause for vertex inconsistency is bidirectional flow with multiple source vertices, although these methods are also useful for correcting poorly-defined ship system data.

After all valid deactivation flow paths have been discovered using the path-finding algorithm, the system database overlays all paths to identify and account for any bidirectional paths within the system and generates a revised adjacency list based on the subset of valid deactivation paths within predefined connectivity between vertices. The new adjacency list is brought into a system evaluation tool for processing the definition of new vertices derived from those original bidirectional-connected vertices.

Figure 3.14 presents an example of a simple deactivation diagram developed from a system that does not require any changes to the network architecture. This layout contains only two unidirectional deactivation flow paths from the source ("D_SYS") to the sink ("A_SYS") and both paths are separate and unique, so no modifications to the system or its vertices are required to derive the deactivation diagram. With only one source in this system, we conclude that all paths for flow can contribute equally towards operation of the sink so the flow convergence at the sink behaves as an "OR" gate in the deactivation diagram.

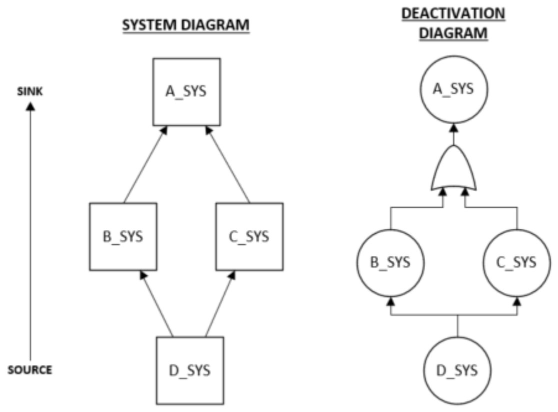


Figure 3.14: Simple Deactivation Diagram

The system evaluation and manipulation process presented here occurs sequentially along each path discovered during the DFS analysis and at each vertex predefined in the system diagram. This process loops over each path to identify required changes within the scope of the path and in respect to all original connections in the system network. Predetermination of changes necessary to maintain system consistency allows for the analysis to operate in a limited scope instead of revaluating the entire system following each incremental change, which is computationally burdensome.

This approach provides improved consistency and scales well to large multiplex systems; however, the methods used require consistent naming schema across all vertices to prevent unintentional redefinition of vertices, aid the reuse of new vertices already created, and remain consistent with other deactivation flow paths. Emulating the RBD structure developed in the ITEM Toolkit by Goodfriend [7], derived system vertices are named using the hyphenated combination of the sink and source names (e.g. "[Sink]-[Source]_SYS") to indicate the basis of each new vertex and the direction of flow. In circumstances where this naming convention is insufficient for completing all connections, the system architecture is supplemented with junction systems that represent auxiliary connections necessary for connecting logic gates between system vertices (e.g. "[Sink]-[Source]-Jct_SYS").

Prior work completed on the VT CR&E process has designated that VCs within the deactivation diagram do not include dependencies for determination of operability. Each VC may be marked as damaged or undamaged during vulnerability analysis for capturing SYS deactivation; however, only SYS's contain sufficient information to detail operability based on deactivation state. SYS dependency data includes a single Boolean logic gate and a list of dependencies containing VCs, other SYS's, or any combination thereof. To combine multiple logic gates into a larger architecture, intermediate junction systems are required as additional vertices to maintain full connectivity between components.

To maintain this structure, each VC should be analytically separate from any vertices that appear before it in the system diagram. System manipulation methods create and insert derived SYS pairs to isolate any VCs that are required to be logically separated from source vertices. These methods also introduce Boolean gates as appropriate for connection of the VC with the deactivation diagram. Figure 3.15 shows an example of this approach to VC autonomy management.

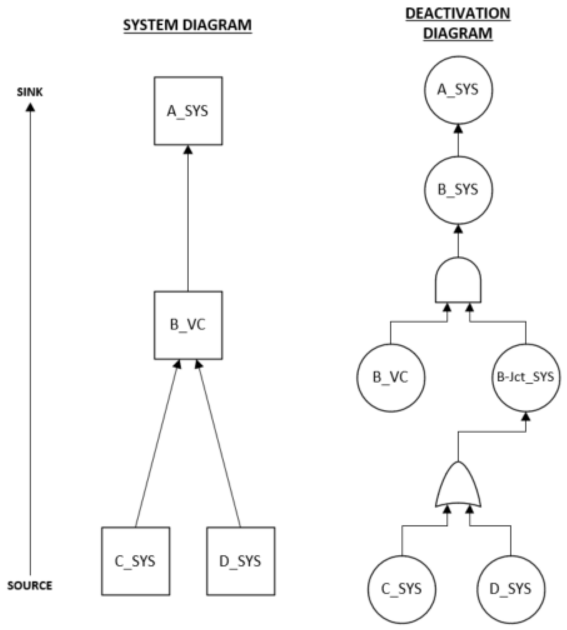


Figure 3.15: Vital Component Deactivation Layout

The design of refined system layouts to ensure that all deactivation flow paths are kept independent requires careful consideration of all localized sources and sinks for each given vertex. Operations affecting each vertex consider the type of vertex (VC/SYS), number of sink and source connections, and how many bidirectional connections currently exist. At the heart of the methodology is a vertex analysis method that contains precomputed structural and behavior variants for 25 unique vertex states determined through network layout and testing.

In bidirectional systems, new vertices are typically required for separating existing vertices into unique flow patterns while maintaining the logical deactivation structure attributed to the original vertices. These changes reflect the necessity of maintaining data lookup for system analysis and retaining pre-existing components and their associated connections. Figure 3.16 shows a simple bidirectional system for deactivation analysis that requires connectivity modifications similar to that demonstrated in Figure 3.15.

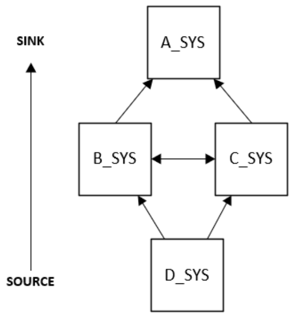


Figure 3.16: Simple Bidirectional System

The example system shown in Figure 3.16 contains four vertices and five connections resulting in four distinct deactivation flow patterns between the source (D_SYS) and the sink (A_SYS). Laying out the unmodified connections between the vertices results in the invalid deactivation diagram in Figure 3.17. Both intermediate vertices, “B_SYS” and “C_SYS”, appear twice in this diagram and have inconsistent network connectivity, producing a data conflict and an invalid deactivation diagram. To correct this error, the bidirectional deactivation paths require introducing additional vertices at junctions to isolate individual vertex contributions to the deactivation analysis and develop unique flow paths. Isolation of “B_SYS” and “C_SYS” using six derived intermediate systems results in the valid deactivation diagram shown below.

The technical design of this approach considers all connections between plexes as part of the total system deactivation analysis and limits each analysis loop to the plex containing to the currently evaluated sink. This methodology prevents massive scaling of analyses across the multiplex system network through connectivity via incompatible flow transfer media. By developing these methods using the object-oriented design approach, native vertices and derived vertices maintain referenceable plex information useful in post-process analysis of individual plexes.

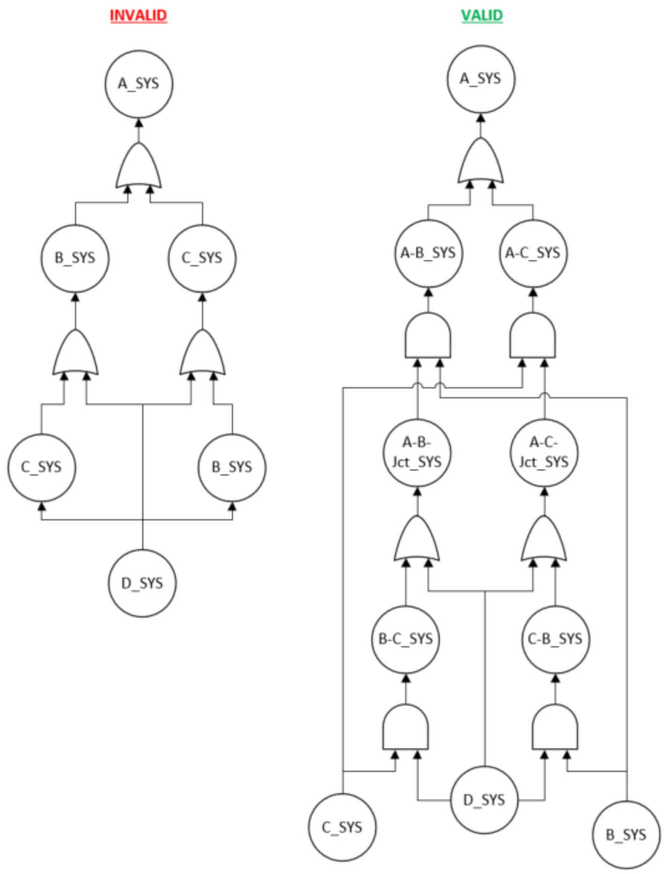


Figure 3.17: Invalid and Valid Deactivation Diagrams from a Bidirectional System

3.4.5 System Validation

As previously described, the system evaluation and manipulation process processes vertices via independent flow paths previously validated using the DFS search algorithm. While processing vertices along the flow paths, new paths containing the original source data and newly-developed vertices are created and stored for data validation and output. System connectivity and adjacency lists/matrices are generated based on the structure of the new path lists instead of the prior analysis data, which enforces consistency of all data and strict validation of all derived connections.

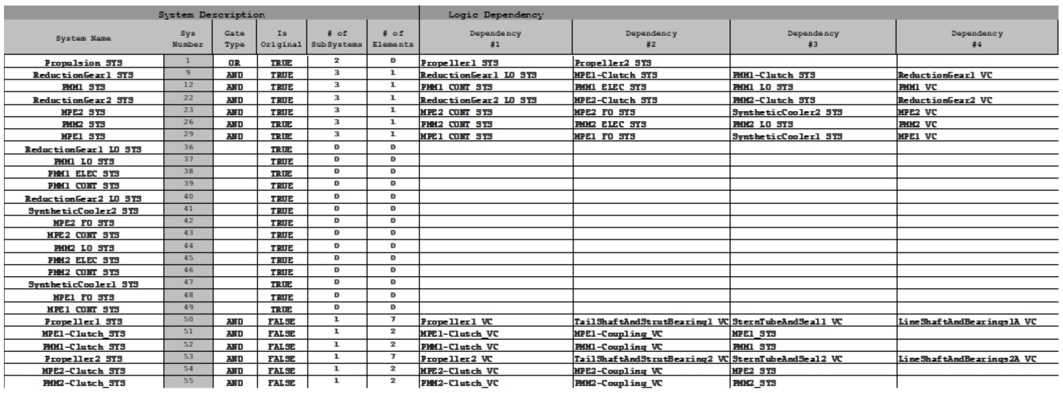
Following export of the new deactivation path list into the system database, the deactivation analysis removes and recreates all arc and edge connections within the database based on the newly-formed paths. A final verification process double-checks that the paths have been well-formed, no duplicate vertices have been unintentionally created, and that the system data is prepared for export.

3.5 Software Utilization

3.5.1 Data Output

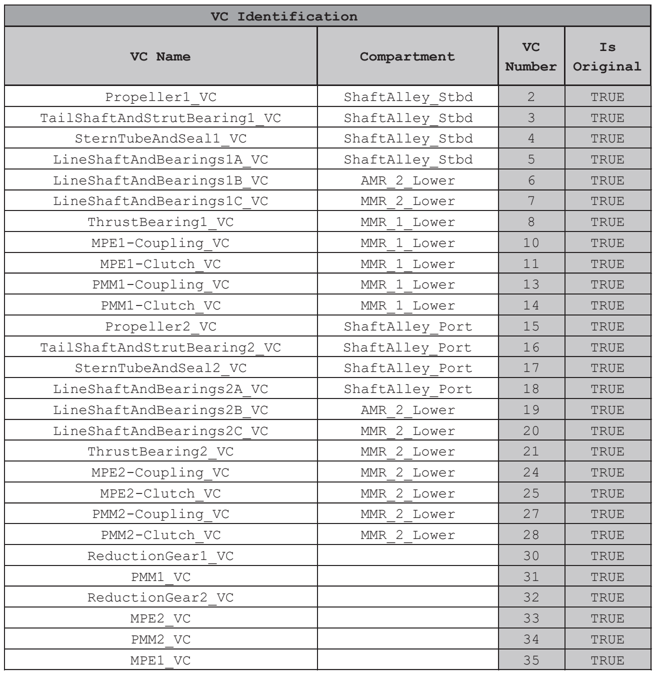
The deactivation analysis tool is built using an object-oriented model approach which stores results of the system deactivation analysis in memory for flexible export capability. The primary export of the tool is into a pair of Excel worksheets consistent with the prior work of Goodfriend [7], “Systems” and “VC”. Examples of each are shown respectively in Figure 3.18 and Figure 3.19. These worksheets store the revised database of systems and vital components after the completed transformation of data into a deactivation form. The data is allotted in a unidirectional adjacency list layout, with each SYS associated with its localized dependencies (sources) and VCs. These worksheets are designed for direct reference by tools built on the VT SSM design framework and other spreadsheet-compatible tools.

The deactivation analysis tool was developed for easy utilization by other researchers working on ship system analysis, so additional export capabilities were designed within the code. The Pajek “.net” file format is built-in with optional data import and export capability, allowing for external file manipulation in text editors and system checking using available network software. Export from Excel to a recognized file format enables one-to-one data validation utilizing the “.net” text-based files and simplified visualization with Pajek.



System Description						Logic Dependency			
System Name	Sys Number	Gate Type	Is Original	# of SubSystems	# of Elements	Dependency #1	Dependency #2	Dependency #3	Dependency #4
Propulsion SYS	1	OR	TRUE	2	0	Propeller1 SYS	Propeller2 SYS		
ReductionGear1 SYS	9	AND	TRUE	3	1	ReductionGear1 LO SYS	MPE1-Clutch SYS	PMM1-Clutch SYS	ReductionGear1 VC
PMM1 SYS	12	AND	TRUE	3	1	PMM1 CONT SYS	PMM1 ELEC SYS	PMM1 LO SYS	PMM1 VC
ReductionGear2 SYS	22	AND	TRUE	3	1	ReductionGear2 LO SYS	MPE2-Clutch SYS	PMM2-Clutch SYS	ReductionGear2 VC
MPE2 SYS	23	AND	TRUE	3	1	MPE2 CONT SYS	MPE2 FO SYS	SyntheticCooler2 SYS	MPE2 VC
PMM2 SYS	26	AND	TRUE	3	1	PMM2 CONT SYS	PMM2 ELEC SYS	PMM2 LO SYS	PMM2 VC
MPE1 SYS	29	AND	TRUE	3	1	MPE1 CONT SYS	MPE1 FO SYS	SyntheticCooler1 SYS	MPE1 VC
ReductionGear1 LO SYS	36		TRUE	0	0				
PMM1 LO SYS	37		TRUE	0	0				
PMM1 ELEC SYS	38		TRUE	0	0				
PMM1 CONT SYS	39		TRUE	0	0				
ReductionGear2 LO SYS	40		TRUE	0	0				
SyntheticCooler2 SYS	41		TRUE	0	0				
MPE2 FO SYS	42		TRUE	0	0				
MPE2 CONT SYS	43		TRUE	0	0				
PMM2 LO SYS	44		TRUE	0	0				
PMM2 ELEC SYS	45		TRUE	0	0				
PMM2 CONT SYS	46		TRUE	0	0				
SyntheticCooler1 SYS	47		TRUE	0	0				
MPE1 FO SYS	48		TRUE	0	0				
MPE1 CONT SYS	49		TRUE	0	0				
Propeller1 SYS	50	AND	FALSE	1	7	Propeller1 VC	TailShaftAndStrutBearing1 VC	SternTubeAndSeal1 VC	LineShaftAndBearings1A VC
MPE1-Clutch_SYS	51	AND	FALSE	1	2	MPE1-Clutch_VC	MPE1-Coupling_VC	MPE1_SYS	
PMM1-Clutch SYS	52	AND	FALSE	1	2	PMM1-Clutch VC	PMM1-Coupling VC	PMM1 SYS	
Propeller2 SYS	53	AND	FALSE	1	7	Propeller2 VC	TailShaftAndStrutBearing2 VC	SternTubeAndSeal2 VC	LineShaftAndBearings2A VC
MPE2-Clutch SYS	54	AND	FALSE	1	2	MPE2-Clutch VC	MPE2-Coupling VC	MPE2 SYS	
PMM2-Clutch_SYS	55	AND	FALSE	1	2	PMM2-Clutch_VC	PMM2-Coupling_VC	PMM2_SYS	

Figure 3.18: Sample SYS Deactivation Data



VC Identification			
VC Name	Compartment	VC Number	Is Original
Propeller1_VC	ShaftAlley_Stbd	2	TRUE
TailShaftAndStrutBearing1_VC	ShaftAlley_Stbd	3	TRUE
SternTubeAndSeal1_VC	ShaftAlley_Stbd	4	TRUE
LineShaftAndBearings1A_VC	ShaftAlley_Stbd	5	TRUE
LineShaftAndBearings1B_VC	AMR_2_Lower	6	TRUE
LineShaftAndBearings1C_VC	MMR_2_Lower	7	TRUE
ThrustBearing1_VC	MMR_1_Lower	8	TRUE
MPE1-Coupling_VC	MMR_1_Lower	10	TRUE
MPE1-Clutch_VC	MMR_1_Lower	11	TRUE
PMM1-Coupling_VC	MMR_1_Lower	13	TRUE
PMM1-Clutch_VC	MMR_1_Lower	14	TRUE
Propeller2_VC	ShaftAlley_Port	15	TRUE
TailShaftAndStrutBearing2_VC	ShaftAlley_Port	16	TRUE
SternTubeAndSeal2_VC	ShaftAlley_Port	17	TRUE
LineShaftAndBearings2A_VC	ShaftAlley_Port	18	TRUE
LineShaftAndBearings2B_VC	AMR_2_Lower	19	TRUE
LineShaftAndBearings2C_VC	MMR_2_Lower	20	TRUE
ThrustBearing2_VC	MMR_2_Lower	21	TRUE
MPE2-Coupling_VC	MMR_2_Lower	24	TRUE
MPE2-Clutch_VC	MMR_2_Lower	25	TRUE
PMM2-Coupling_VC	MMR_2_Lower	27	TRUE
PMM2-Clutch_VC	MMR_2_Lower	28	TRUE
ReductionGear1_VC		30	TRUE
PMM1_VC		31	TRUE
ReductionGear2_VC		32	TRUE
MPE2_VC		33	TRUE
PMM2_VC		34	TRUE
MPE1_VC		35	TRUE

Figure 3.19: Sample VC Deactivation Data

3.5.2 Visualization

Validation of network analysis has the potential to be a significantly laborious task, even with relatively small systems. For the purposes of validation and report generation, visualization plays a key role in developing a summarized output of the deactivation diagram tool. Visualization capability development for the deactivation analysis tool centered on the use of Pajek and Microsoft Visio.

Pajek is a powerful network analysis tool, useful for 3-D directed graph visualization and qualitative analysis of large networks. The Pajek ".net" data format previously described provides simple end-user analysis and modification along with interoperability with similar network tools. Visualization of vertices in Pajek requires only a well-defined input file, as represented in Figure 3.11. In addition, the physical locations of vertices may be defined within Excel and included in the ".net" output for representation in the Pajek network diagram. For vertices missing locational data, the layout may be manually distributed or automatically created using algorithm-based layout distributions.

Pajek lacks the capability for displaying Boolean gates necessary for creating a deactivation diagram but may be used for visual comparison of changes to network architecture resulting from this analysis. Figure 3.20 and Figure 3.21 show network diagrams generated in Pajek for a chilled water (CW) system before and after completion of the deactivation analysis process. Close inspection of the diagram in Figure 3.21 shows all of the original vertices (in black) and the additional vertices (in grey) added to the system to ensure unidirectional system flow and structural data consistency.

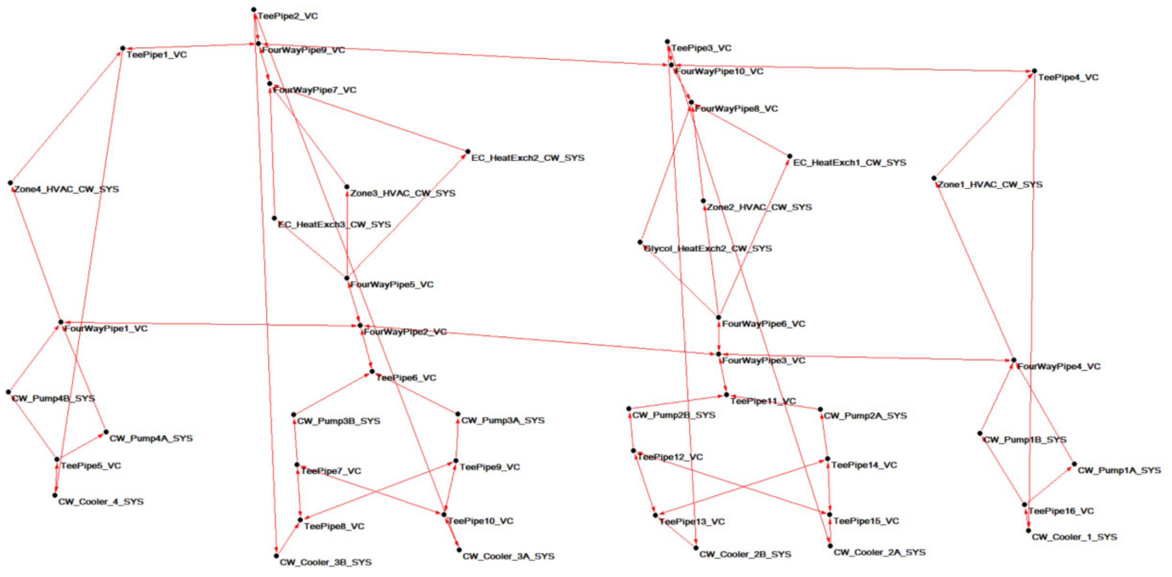


Figure 3.20: Chilled Water Source Network Diagram in Pajek

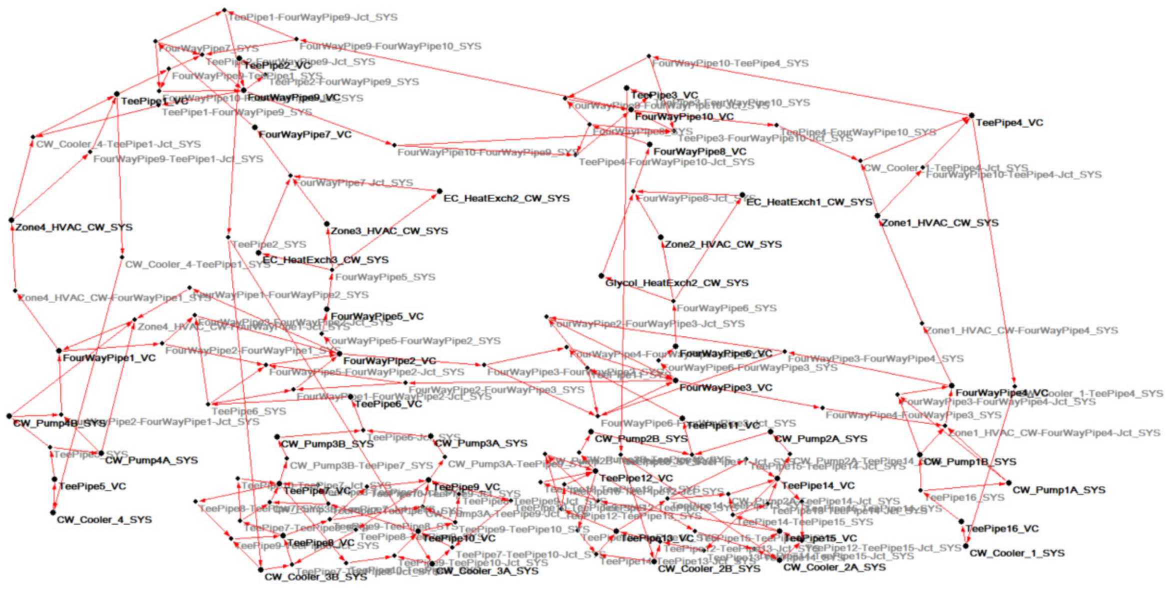


Figure 3.21: Chilled Water Deactivation Network Diagram in Pajek

As part of the Microsoft Office family, Visio incorporates native VBA support and provides useful tools for developing network and deactivation diagrams. With the use of VBA, deactivation diagrams can be developed directly by export methods built within the Excel-based system deactivation analysis tool and launched in Visio. Visualization within Visio uses the Fault Tree Analysis Diagram template from Microsoft to ensure consistency of deactivation diagram tree creation. Unlike visualization in Pajek, Visio objects exist only in a 2D workspace and require extensive manipulation in-code to form all connections and develop a clean diagram as shown in Figure 3.22. Visio diagram creation is powered in Excel VBA.

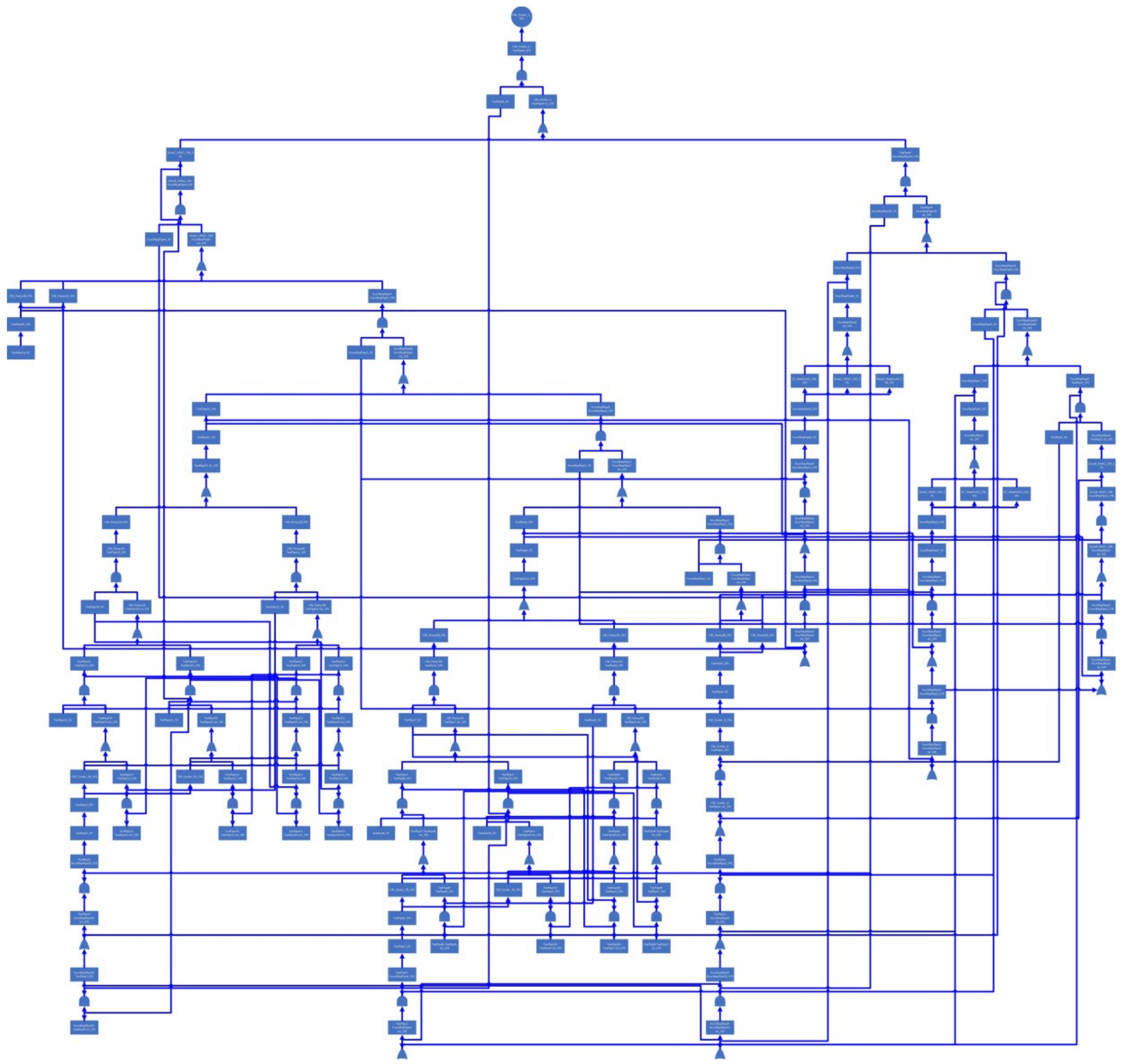


Figure 3.22: Chilled Water Deactivation Diagram in Visio [4]

Methods for Visio visualization begin with the vertex adjacency list developed in the deactivation analysis and create individual diagrams for each sink within the system. This adjacency list is piped through a Breadth-First Search (BFS) algorithm to create deactivation diagrams that sequentially begin and branch out from each sink. The process uses a breadth-first approach instead of depth-first to maintain consistency and clarity of the resultant diagrams. Internal queue data objects retain vertex data information in the BFS algorithm for creating path links between vertices.

As Visio exists as a separate entity from Excel, the software libraries necessary for Visio export are located and loaded at compile-time within the Excel-based code. Visio export is limited to computers with Visio installed. Weakly-typed library objects permit the export operation to fail quietly on systems without Visio. The code has only been validated using Visio 2016 and 2019, other versions of Visio may be partially or fully non-compatible.

3.5.3 Data Filtering

Data filtering is a high priority item for the study of data from deactivation analysis. The design of the tool for correlation with the preexisting vulnerability work evolving at VT resulted in a unified ship system output in Excel; however, this amount of data is often too much for visualization and other consumers. Data analyzed by the system deactivation tool may come from a large database containing a complete multi-plex system, so the decision was made to incorporate post-process data filtering to allow for fine-tuned analysis of ship systems.

These filters, built into the input GUI for the code, limit the plexes that are output by the tool to Excel and any other data consumers. On the individual level of each input data file or worksheet, the user can define whether to output all plexes or limit the scope of output accordingly. The determination for making this change was to support localized analysis of singular plexes while maintaining all system characteristics of the fully-appended deactivation diagram model.

Due to the intended filter use to have no impact on the results of the analysis, the analysis process was not to be modified by the operation of the filter. Investigations performed into the possibility of trimming the input data during filter usage indicated high likelihood for unintended consequences of data corruption or invalid/incomplete analysis when modifying the input data. Operation of the filter was therefore developed to occur exclusively within post-processing, so it provides no reduction in analysis time and a negligible increase in computational time during data export.

3.5.4 LEAPS Integration

During the development of the code, it was proposed that this work be integrated with the U.S. Navy's Leading-Edge Architecture for Prototyping Systems (LEAPS) [11] data repository for future analysis of ship systems built using LEAPS tools (e.g. LEAPS Editor, S3D) within the structure of the Formal Object Classification for Understanding Ships (FOCUS) Product Meta-Model (PMM) framework.

The FOCUS PMM is an evolving framework designed to keep a strict structure of ship-related data and is closely managed by NSWCCD to ensure that new capabilities are consistent with broad expansion plans. Information may be stored in a LEAPS ship database outside of the FOCUS PMM; however, such data is not evidently available to other software programs operating on the same database without preexisting knowledge of its uncompliant data structure. The Smart Ship System Design (S3D) tool, currently under development by the Electric Ship Research and Development Consortium (ESRDC) and referenced in this paper, is an example of a tool that required storage of data beyond current FOCUS definitions; the FOCUS PMM is in the process of being extended to accommodate these additional needs through the creation of additional component types and properties.

The LEAPS API is an OS-independent C++ library, so all interfacing tools must be built to interface directly with the libraries or wrap them for use in a different language. A full wrap of the LEAPS API for VBA would be a significantly custom code and require regular updating, so it was left outside of the scope of this work. As an alternative, subsequent work performed with Chalfant and Parsons [4] has resulted in the creation of a data export code built upon ongoing work of Chalfant [12], named the LEAPS Network Translator. This code uses the Pajek ".net" file format already incorporated into the deactivation analysis tool to export data for ingest into the Excel-based tool.

Currently, neither S3D nor FOCUS have set guidelines for analysis of system flow direction, so this code was built accounting for those limitations. While recommendations for future expansion of FOCUS to incorporate flow directionality are being reviewed, the LEAPS Network Translator code uses explicit component type analysis to apply flow properties to the S3D ship system model, such as known fluid flow direction through a pump or heat exchanger. With these explicit assumptions applied, each specified plex is evaluated for connectivity, system type (open/closed loop), and sink/source identification through a regression analysis.

3.6 Applications

Work described in this paper has sought to develop an approach to naval ship system deactivation analysis that can be incorporated within existing and future tools developed as part of ongoing work on preliminary distributed system design and analysis in early-stage ship design. This approach seeks to fill the gap in developing whole-ship deactivation analyses at Virginia Tech and provide a capability for analyzing large multiplex ship systems, such as the integrated multi-plex mechanical propulsion system shown in Figure 3.23.

Large ship systems often present a significant barrier to efficient deactivation and flow analysis. Preliminary design efforts, such as the VT C&RE process, require agile processing of system connectivity to incorporate derived analyses (e.g. vulnerability, operability) effectively into practical design processes. These efforts have proven that a deactivation analysis can be significantly useful in precomputing data for analyses that would otherwise be stalled by slow-running flow analysis.

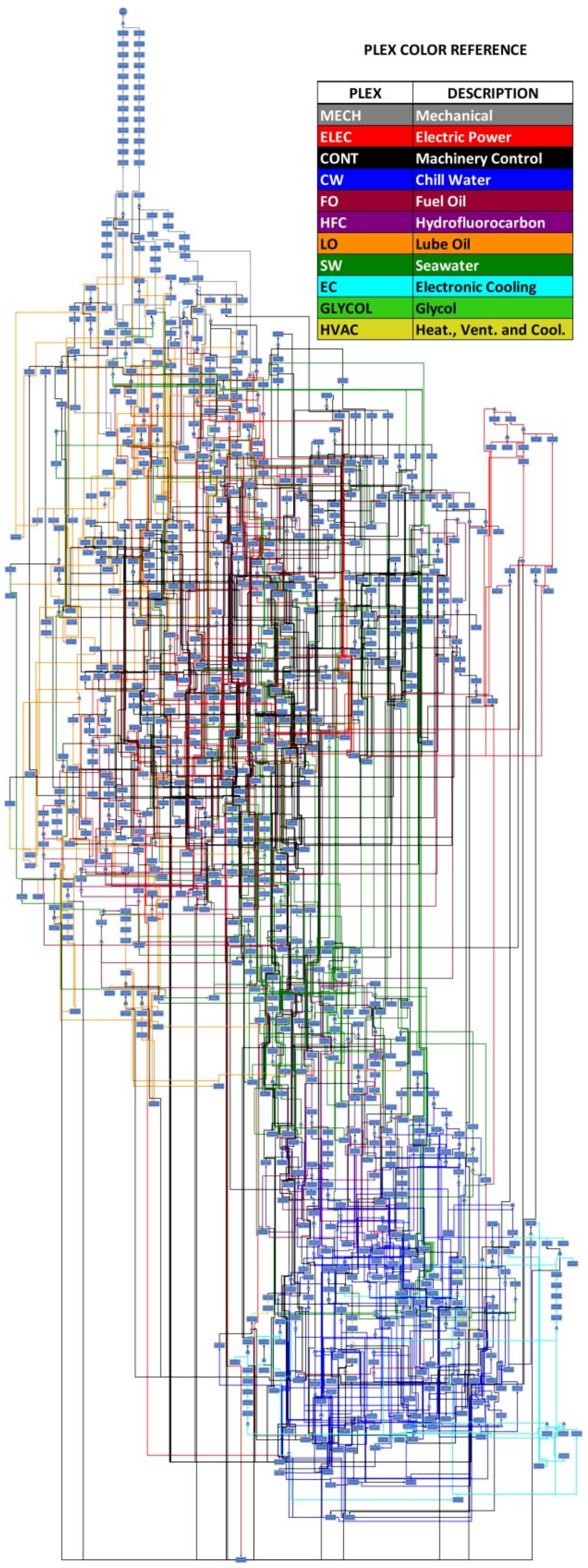


Figure 3.23: Mechanical Propulsion Total Deactivation Diagram [4]

3.7 Limitations

3.7.1 System Descriptions

As previously described with regard to path-finding, computational processing of ship systems requires a description of systems that is strictly bounded and well-defined within the data source. In a large distributed system network, the logical beginning and end for flow diagrams is not computationally identifiable without a degree of additional system information, such as flow direction through certain components or energy transfer in or out of a plex via mechanical, electrical, or thermal interfaces. With sufficient data, system extents and flow direction can be implicitly evaluated such as completed by the LEAPS Network Translator tool; however, without information provided by the system object type association in S3D, input data originating in Excel requires explicit identification of overall plex sinks and sources to trace out system flows.

The development of well-defined input data has been highlighted as difficult in the VT SSM process by this work on system deactivation analysis. To best manage and identify poorly-defined system descriptions and missing data, work on this code has incorporated a significant number of validation checks. These data checks analyze the input data first for conformity to expected layout and identification schema, including identifying overlapping or duplicate data, then run progressive checks on the system as a whole as it undergoes evaluation. For a minimal increase in computational time, these error checks lend a significantly greater degree of confidence in result accuracy.

3.7.2 VBA Restrictions

While Visual Basic for Applications (VBA) is widely-distributed through its inclusion within Microsoft Office products and has many users, it remains a non-robust programming language and faces some significant performance limitations imposed by its runtime environment and limited expandability. VBA is a single-threaded programming language and cannot incorporate multi-threaded optimization of workloads, which hinders full utilization of computational performance in large system analysis. In addition, the memory allocation limit of 2 GB in the 32-bit version of Excel found on most consumer computers requires careful consideration of analysis memory overhead and requires data consolidation to not exceed runtime memory limits. To support the widest range of users, the code developed in this work was designed for compatibility with 32-bit Microsoft Excel and was carefully optimized to remain within the lower memory limits of the software.

Thorough performance testing performed during the development of this paper has indicated poor optimization and inconsistent performance of the VBA Collection data type, in some cases performing up to 50x slower than array operations when Collection object keys are not explicitly defined. On a large multiplex system analysis, such hinderances cripple the runtime performance of the code and had previously made it unusable for the VT C&RE optimization process. Changes to the code to avoid performance bottlenecks and caching lightweight data lookup objects in lieu of progressive searching have significantly sped up the code operation. On a mid-tier consumer laptop, system deactivation analyses without data export currently take approximately one minute to run for the VT multiplex ship system model. Earlier analyses (non-optimized) had runtimes of several hours.

The poor computational speed of VBA and limited object types from standard libraries are detrimental to large analyses, so future work may include re-working the source into C++/VB. At this time, the most significant slowdown is in the optional Visio data export. Multiplex system export with no filters enabled has been shown to take overnight to complete, with an Excel-to-Visio processing speed of 1-2 seconds per vertex. Optimization of the visualization export was outside the scope of this work; however, continued code optimization or the use of an externally-compiled library would likely greatly improve visualization throughput.

3.8 Conclusions

The deactivation analysis methodology presented in this paper offers support to continued advancement in the development of methods and tools for ship system analysis. This paper ties together a network architecture framework with over 20 years of development at Virginia Tech with specific techniques for system deactivation analysis and demonstrates how these techniques may be useful for existing and future toolsets.

Results of this analysis were only previously available to researchers at Virginia tech as narrowly-scoped examples of deactivation diagrams derived from manual system analysis. This work has provided new capabilities for application of automated analysis towards ship systems and the enablement of advanced derived analyses based on this work. Analyses poised for improvements based on this work include naval ship system vulnerability [7], flow optimization [5], and architectural framework analysis [4].

Tools presented in this paper for automated naval ship system deactivation analysis have gone through numerous stages of development and testing to meet strict targets for overall capability. They have been designed to be stable and robust and have pushed the limits of computational throughput in Excel VBA. These methods and approach provide added value and computational capabilities to existing tools in the Virginia Tech C&RE process and open up new paths for future analysis software development.

These developments were only possible through ship system diagram development in conjunction with fellow Virginia Tech researchers and the support of outside research groups working on the LEAPS database.

3.9 Future Work

System deactivation analysis and deactivation diagram development require a consistent and well-structured source data set. Information that cannot be interpreted from a typical adjacency matrix, such as system sinks/loads and interconnectivity between plexes, must be explicitly defined within input data developed by hand or through an analysis tool such as the LEAPS Network Translator to develop meaningful diagrams. This deactivation diagram tool contains a significant number of data consistency and validation checks; however, small errors in the source data may result in drastic changes to the final deactivation diagram system configuration. Future efforts shall focus on improving validation of input data to ensure accurate and consistent results.

The methods presented here were developed to directly interface with and support Virginia Tech's ongoing research using spreadsheet analysis and retain ease of maintainability for future researchers, so VBA was the choice programming language for this initial effort. With a proven methodology based on this initial work, future transformation of the tools presented in this paper into a non-embedded language, such as C++, VB, or Python, has the potential for significantly greater computational performance than the current manifestation of this methodology. These additional efforts would significantly improve the capabilities shown in this paper.

3.10 References

[1] W. R. Merz, "OPNAV Instruction 9070.1B," Department Of The Navy, Washington DC, 2017.

[2] D. Brefort *et al.*, "An Architectural Framework For Distributed Naval Ship Systems," *Ocean Engineering,* vol. 147, pp. 375-385, 1 Jan. 2018.

[3] M. A. Parsons *et al.*, "Application of a Distributed System Architecture Framework to Naval Ship Concept and Requirements Exploration (C&RE)," in *Proc. ASNE Intelligent Ship Symposium*, 2019, in press.

[4] D. J. Snyder, M. A. Parsons, A. J. Brown, and J. Chalfant, "Network Architecture Framework Applications with FOCUS-Compliant Ship Designs," in *Proc. 2019 IEEE Electric Ship Technologies Symposium (ESTS)*, in press.

[5] M. A. Parsons *et al.*, "Early-Stage Naval Ship Distributed System Design Using Architecture Flow Optimization," *Naval Engineers Journal*, unpublished.

[6] A. J. Brown, "Ship and Marine Engineering System Concept Design," in *Marine Engineering*, M. G. Parsons, Ed. 4th ed. Alexandria, VA: SNAME, in press, pp. 1-39.

[7] D. B. Goodfriend, "Exploration of System Vulnerability in Naval Ship Concept Design," M.S. Thesis, Department of Aerospace and Ocean Engineering, Virginia Polytechnic and State University, Blacksburg, VA, 2015.

[8] C. A. Ericson II, "Fault Tree Analysis - A History," in *Proc. 17th International System Safety Conference*, Orlando, FL, 1999: System Safety Society.

[9] D. B. Goodfriend and A. J. Brown, "Exploration of System Vulnerability in Naval Ship Concept Design," *Journal of Ship Production & Design,* Article vol. 34, no. 1, pp. 42-58, 2018.

[10] T. H. Cormen, *Introduction To Algorithms*, 3rd ed. Cambridge, Mass.: MIT Press, 2009, pp. xix, 1292 p.

[11] Naval Surface Warfare Center Design Tool Development Branch, *LEAPS Editor User's Manual Version 5.0*. West Bethesda, MD: Naval Surface Warfare Center Carderock Division, 2015. User's Manual available with LEAPS distribution.

[12] J. Chalfant, Z. Wang, and M. Triantafyllou, "Expanding the Design Space Explored by S3D," in *Proc. 2019 IEEE Electric Ship Technologies Symposium (ESTS)*, in press.

Chapter 4

Conclusions and Future Work

This thesis details the development of a refined framework and methodology for ship system deactivation analysis with a specific focus on deactivation diagrams and associated applications towards vulnerability analysis. The two manuscripts included in this thesis have demonstrated new deactivation analysis capabilities with complementary software tools targeting LEAPS ship system definitions and the Virginia Tech network architecture framework respectively.

Software development undertaken as a part of this thesis has resulted in the generation of over 7,000 lines of mixed VBA and C++ code. These software tools and the operational methods contained within provide a solid code base for operation as-is and for future expandability.

4.1 Conclusions

Testing of the analysis methodologies and software tools discussed in this thesis has been scaled across a wide range of system data, validating these methods against sample data sets and large-scale multiplex ship systems. Results from these computational methods are consistent and have been corroborated by hand calculations and expected system architectures.

Tools developed in support of this thesis are intended to provide significantly useful contributions to ongoing research work in ship concept design and design optimization. Work presented in this thesis is tentatively anticipated to deliver the following benefits to future research:

- A methodology for ship system deactivation analysis that is well-founded in network theory and general system architecture. Results from this methodology have been tested and validated against detailed partial and complete ship systems.
- The Deactivation Analysis Tool, which was developed in Excel VBA and designed for direct incorporation into the Virginia Tech Ship Synthesis Model. This tool contains useful functions to handle automated data import/export, deactivation analysis, and deactivation diagram visualization.
- Modularized code for ship system description export from S3D-developed LEAPS databases into the Deactivation Analysis Tool for use in deactivation diagram development and system architecture analyses at Virginia Tech. This code was developed with the direct support of Dr. Julie Chalfant and has been incorporated with her ongoing work.
- Supplemental I/O validation routines and "helper" functions developed through extensive code optimization and data analysis efforts. These routines may help to support and advance future related software development efforts.

Ship deactivation analyses performed in validation of this work have provided a scope of the complexity of the challenge in redefining complex ship systems into the constraints of a deactivation diagram form, as demonstrated by the multiplex architecture in Figure 4.1. Early indications show that the deactivation diagram approach to simplifying ship system analyses currently provides the best technique for accelerating large-scale computational work such as vulnerability analysis; however, future work is expected to develop faster and more agile techniques to multi-state system analysis. These techniques should continue to be developed and pushed forward to further advance the state of the art.

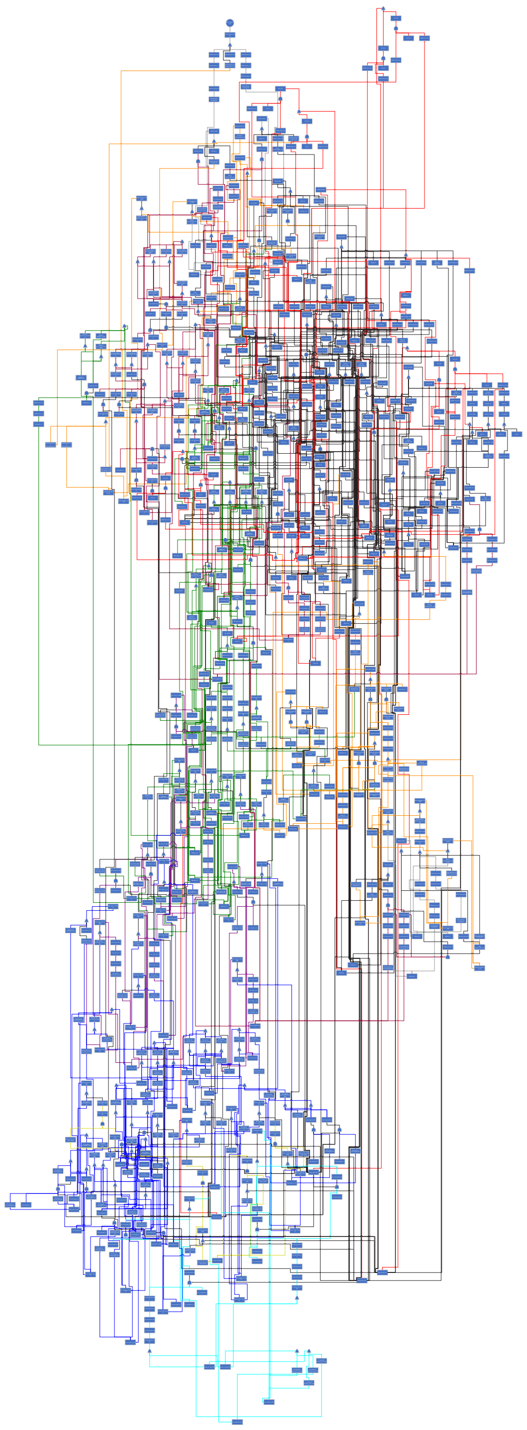


Figure 4.1: Multi-Plex Total Ship Deactivation Diagram

4.2 Future Work

Work presented here has opened up new opportunities for future efforts and analyses built on the principles described in this thesis. These opportunities include the following:

4.2.1 Software Upkeep/Advancement

Software tools described in this thesis have been developed to be easily accessible, modular, and maintainable by other researchers with moderate levels of experience in code development. As the Naval Ship C&RE process at VT continues to develop and changes are made to the current SSM definitions for Combat, Power and Energy Systems (CPES), code developed as part of this thesis will require upkeep and advancement to stay current with state-of-the-art analyses.

4.2.2 LEAPS Integration

Initial efforts as part of this thesis in the merging of LEAPS database data into the VT SSM have enabled the extraction of ship system architecture data for deactivation analysis in the LEAPS Network Translator. These efforts relied on preexisting knowledge of the definitions of ship systems in S3D outside of the FOCUS PMM and do not fully encapsulate the entirety of available system definitions through LEAPS. As the FOCUS PMM is updated to include enhanced ship system detail information, future work should focus on complete LEAPS integration with the VT C&RE process.

4.2.3 Optimized Data Analysis

The Excel-based Deactivation Analysis Tool was designed on the concept of easy accessibility to most researchers. To improve computational performance and add to its capabilities, the code may be redeveloped in a non-embedded language with improved data efficiency and software parallelization. The techniques for analyzing ship system connectivity using recursive algorithms and isolating the individual analysis of each vertex are expected to be portable to a parallelized computational environment not currently available within the limits of VBA programming.

4.2.4 Real-Time System Deactivation Analysis

Developments in large-scale computing and machine learning are expected to allow for the real-time analysis of system connectivity, flow, and deactivation. New techniques for processing system data may yield alternatives to the precomputation of deactivation diagrams and greatly improve system architectural analyses.

Appendix A

Deactivation Analysis Tool Structure

The programming of the Deactivation Analysis Tool was undertaken as an object-oriented software development project. This structure allows for flexibility in design, concise interactions between data objects, and a robust data structure. The following breakdown of the data structure provides a clear view of the development framework and an outline of the design methodology.

The Excel VBA code development structure separates software code into the following four categories: Microsoft Excel Objects, Forms, Modules, and Class Modules. The majority of code development in this thesis is in class modules, each encapsulating a custom object that contains internal data and functions targeted specifically at a bounded capability set. Each class module "object" in this object-oriented approach has its own set of roles in the analysis process and has been developed to fit within the logical structure of the deactivation process.

Folder structures for organizing software code are common in other languages but do not exist in VBA. Readability and maintainability are a priority in the development of this tool, so the naming conventions for modules and variables reflects the intention to improve organization and clarity of the code.

The following software architecture was developed in VBA as part of this effort:

Forms:

- ProgressForm
    - Form for the display of status outputs from the deactivation analysis. This provides direct feedback on present state and elapsed computation time.

Modules:

- DeactivationAnalysisMain
    - Primary module for execution of the deactivation analysis code. This module links together sequential operation of the tool class modules for analysis execution and manages system data.
- UtilityDataTypes
    - Collection of global custom data types and enumerations used within the class modules.
- UtilityTools
    - Collection of standalone procedures used to support deactivation analyses. The static functions and subroutines in UtilityTools perform repetitive tasks (e.g. string manipulation and array modification) and ensure consistency across the analysis.

Class Modules:

- DataObj_Arc
    - Object representation of a unidirectional connection between two system vertices, containing explicit identification of the parent and child vertices by vertex index.
- DataObj_Edge
    - Object representation of a bidirectional connection between two system vertices, containing explicit identification of both vertices by vertex index.
- DataObj_System
    - The most complex data object in the deactivation analysis code. This class module contains the full ship system definition being processed by the deactivation analysis and provides a consistent data repository for passing between (class) modules. The system data object contains internal collections of other class modules, including vertex lists and arc/edge connections, and has internal procedures for merging systems and data validation.
- DataObj_SystemConn
    - Object representation of an architectural system connection within a plex (explicit) or between different plexes (implicit). This object provides the search parameters for performing path-finding analysis.
- DataObj_Vertex
    - Object representation of a network vertex and associated properties. This object contains localized data necessary for maintaining network connections and consistency, including gate connection type, directionality of connections (unidirectional/bidirectional), and special vertex object types (e.g. source, sink).
- DataObj_VertexColl
    - Collection object specifically developed for containing vertex objects and providing rapid lookup of vertices using cached vertex properties.
- DataObj_Visio
    - Object representation of a basic Visio graph object relationship, used for caching data necessary for making graph object connections in the Visio Breadth-First Search (BFS) export algorithm.
- SysObj_Dictionary
    - Wrapper for the Microsoft System Dictionary object, a key-value pair collection object.
- SysObj_Queue
    - Wrapper for the Microsoft System Queue object, a first-in, first-out (FIFO) collection object

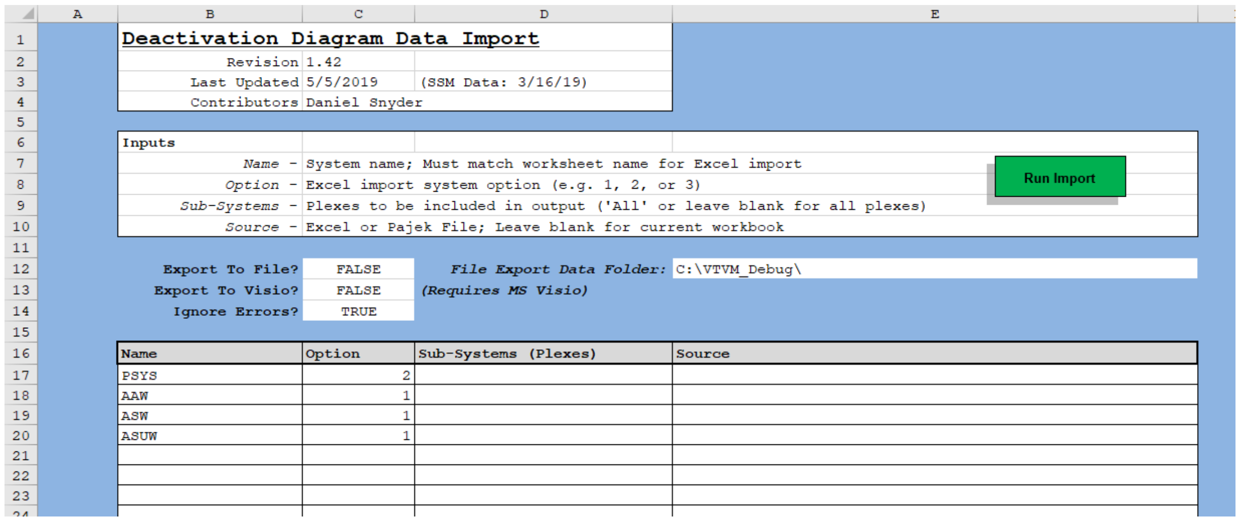
- SysObj_Stack
    - Wrapper for the Microsoft System Stack object, a last-in, first-out (LIFO) collection object.
- Tool_RecursiveDFS
    - Tool developed for performing a path-finding algorithm based on the Depth-First Search (DFS) algorithm across the system description read into the deactivation analysis. The resultant collection of identified paths is post-processed by "Tool_SystemEval."
- Tool_SystemEval
    - Tool for the redefinition of system vertices and structure into a unidirectional deactivation diagram system representation. This tool uses the collection of paths from the DFS analysis to develop the alternative system view. The logic for developing new vertices to suit the deactivation diagram is contained within this tool.
- Tool_SystemIO
    - Tool containing a collection of functions necessary for importing and exporting text-based system architecture and deactivation data from Excel and external ".net" Pajek-style files.
- Tool_VisioExport
    - Tool developed to instantiate Visio and create a visual representation of the post-analysis deactivation diagram(s). This tool contains all of the functions necessary to parse the system data and develop deactivation diagrams using fault tree templates in Visio.

Appendix B

Deactivation Analysis Tool Operation

B.1 User Interface

Configuration and operation of the deactivation analysis tool occurs through the standard Excel user interface, shown in Figure B.1. Manual tool execution is triggered by the “Run Input” button, which calls the Excel macro “DeactivationAnalysis_Run”. Automated processes using this tool may call the execution macro directly from VBA code.



Deactivation Diagram Data Import

Revision	1.42
Last Updated	5/5/2019 (SSM Data: 3/16/19)
Contributors	Daniel Snyder

Inputs

Name -	System name; Must match worksheet name for Excel import
Option -	Excel import system option (e.g. 1, 2, or 3)
Sub-Systems -	Plexes to be included in output ('All' or leave blank for all plexes)
Source -	Excel or Pajek File; Leave blank for current workbook

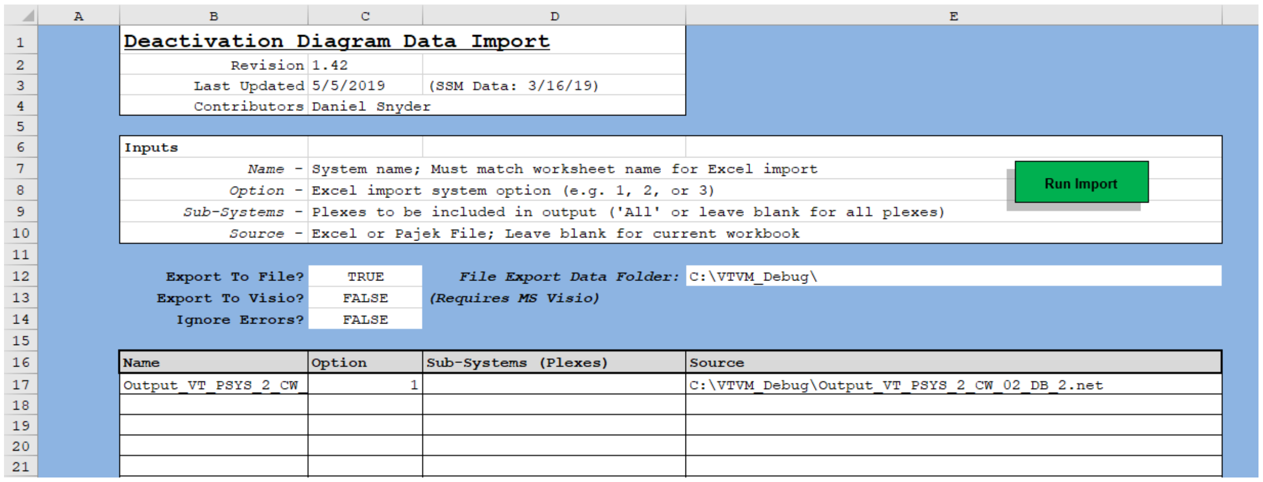
Export To File?	FALSE	***File Export Data Folder:***	C:\VTVM_Debug\
Export To Visio?	FALSE	***(Requires MS Visio)***	
Ignore Errors?	TRUE		

Name	Option	Sub-Systems (Plexes)	Source
PSYS	2		
AAW	1		
ASW	1		
ASUW	1		

Figure B.1: Analysis User Interface for Internal Data

The analysis methodology is designed to handle system architecture input from both Excel data worksheets and external “.net” data files. Figure B.1 shows an example of a prepared input form developed for internal Excel data input. Figure B.2 demonstrates the input of an external “.net” data file. Analysis is constrained only by the runtime memory requirements and data allocation thresholds of Excel VBA, there are no hard limits imposed on the number of input data sources for the tool. Both input source types (Excel & “.net”) may be used concurrently provided that the system data does not conflict between the multiple sources.

Where data conflicts do occur, or invalid data is encountered, data consistency checks within the deactivation analysis code provide resiliency against hard failures of the program. If the input data has a conflicting definition or has been developed in a way unaccounted-for in the code, a pop-up message will appear and request the user’s input on whether the process should continue or not, as shown in Figure B.3.



Deactivation Diagram Data Import

Revision	1.42
Last Updated	5/5/2019 (SSM Data: 3/16/19)
Contributors	Daniel Snyder

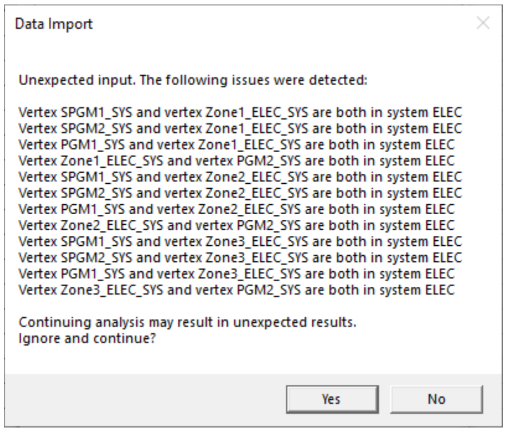
Inputs

Name -	System name; Must match worksheet name for Excel import
Option -	Excel import system option (e.g. 1, 2, or 3)
Sub-Systems -	Plexes to be included in output ('All' or leave blank for all plexes)
Source -	Excel or Pajek File; Leave blank for current workbook

Export To File?	TRUE	***File Export Data Folder:***	C:\VTVM_Debug\
Export To Visio?	FALSE	***(Requires MS Visio)***	
Ignore Errors?	FALSE		

Name	Option	Sub-Systems (Plexes)	Source
Output_VT_PSYS_2_CW	1		C:\VTVM_Debug\Output_VT_PSYS_2_CW_02_DB_2.net

Figure B.2: Analysis User Interface for External Data



Data Import

Unexpected input. The following issues were detected:

Vertex SPGM1_SYS and vertex Zone1_ELEC_SYS are both in system ELEC
Vertex SPGM2_SYS and vertex Zone1_ELEC_SYS are both in system ELEC
Vertex PGM1_SYS and vertex Zone1_ELEC_SYS are both in system ELEC
Vertex Zone1_ELEC_SYS and vertex PGM2_SYS are both in system ELEC
Vertex SPGM1_SYS and vertex Zone2_ELEC_SYS are both in system ELEC
Vertex SPGM2_SYS and vertex Zone2_ELEC_SYS are both in system ELEC
Vertex PGM1_SYS and vertex Zone2_ELEC_SYS are both in system ELEC
Vertex Zone2_ELEC_SYS and vertex PGM2_SYS are both in system ELEC
Vertex SPGM1_SYS and vertex Zone3_ELEC_SYS are both in system ELEC
Vertex SPGM2_SYS and vertex Zone3_ELEC_SYS are both in system ELEC
Vertex PGM1_SYS and vertex Zone3_ELEC_SYS are both in system ELEC
Vertex Zone3_ELEC_SYS and vertex PGM2_SYS are both in system ELEC

Continuing analysis may result in unexpected results.
Ignore and continue?

Figure B.3: Unexpected Input Warning Message

The deactivation analysis tool spreadsheet contains a number of fields for fine-tuning the analysis. The primary fields are the source data input fields and are as follows:

<u>Name</u>

- This is the system name. When utilizing an Excel spreadsheet import data source, this must exactly match the spreadsheet name for data import. This value is specified by the user for ".net" file data import.

<u>Option</u>

- When utilizing an Excel spreadsheet import data source, this is for the selection of a system option number (e.g. 1,2,3,…) based on the VT Ship Synthesis Model (SSM) data layout. Only one value may be specified.

Sub-Systems (Plexes)

- When utilizing an Excel spreadsheet import data source, this is for explicit selection of individual plexes to include in the analysis. Plexes are identified by name and concatenated with a comma or semi-colon. Leaving this field blank or typing in "All" will include all plexes in the analysis.

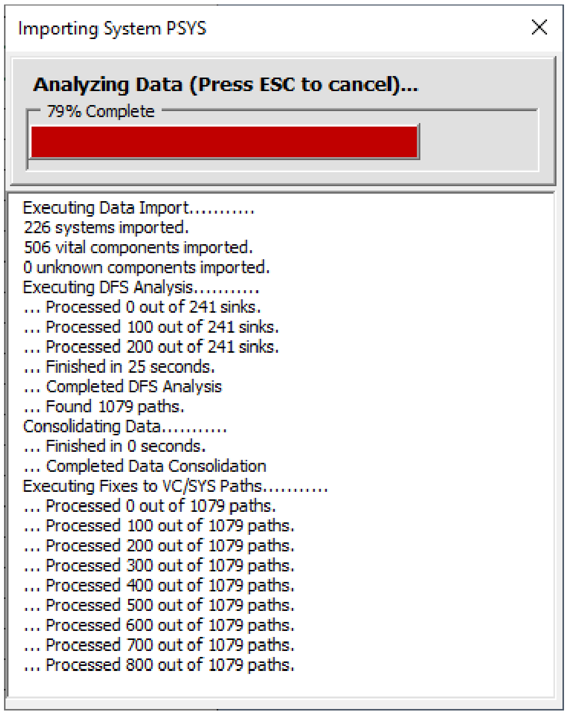
Source

- When utilizing a ".net" import data source, this field points the tool towards the location of the data file. This must be accurate and the user must have permissions to access the file location.

A secondary set of fields allows the user to enable output from the analysis to a ".net" Pajek-type output file and to Visio. The former option, when enabled, will generate a set of output files that represent the system data prior to the deactivation analysis and following it. This has been largely developed for data checking but can also be used for data export and visualization. The latter option will export to Visio if the user's machine has Visio installed. This has been tested to work with Visio 2016 and 2019. Visio export is a significantly slower option for export with large systems sometimes taking a few hours to complete export. The user is advised that to reduce the chance of errors or crashes that the machine be left alone until export is completed. The "Ignore Errors" field is to disable all warning messages and indicate that the analysis should proceed automatically.

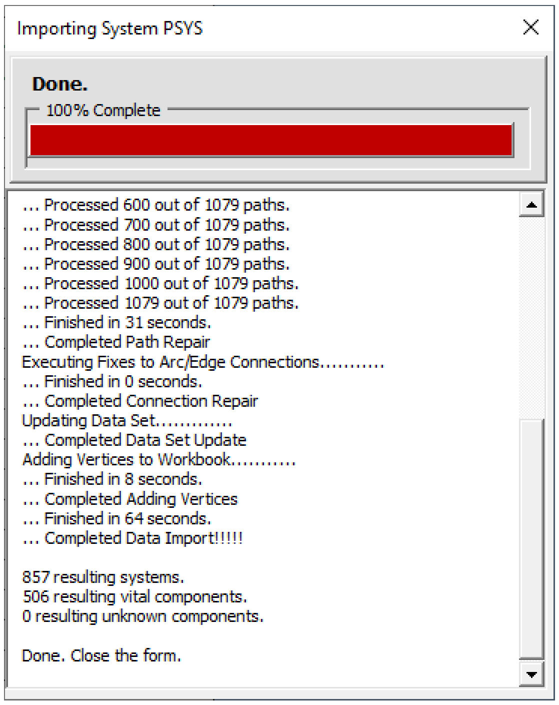
While deactivation analysis operations are ongoing, a status form will indicate the current state of the analysis, shown in Figure B.4. Once the full operation has completed, including data export, the status form will display "Done" and "100% Complete", shown in Figure B.5, as well as a summary of system modifications for deactivation and the total runtime of the analysis.

If the analysis needs to be cancelled at any point, the "Esc" key has been programmed to attempt to halt the process. Since VBA is a single-threaded language, this may or may not stop the process; however, it is recommended that the user first try pressing the "Esc" key before forcing Excel to close via the Windows Task Manager.



```
Executing Data Import...........
226 systems imported.
506 vital components imported.
0 unknown components imported.
Executing DFS Analysis...........
... Processed 0 out of 241 sinks.
... Processed 100 out of 241 sinks.
... Processed 200 out of 241 sinks.
... Finished in 25 seconds.
... Completed DFS Analysis
... Found 1079 paths.
Consolidating Data...........
... Finished in 0 seconds.
... Completed Data Consolidation
Executing Fixes to VC/SYS Paths...........
... Processed 0 out of 1079 paths.
... Processed 100 out of 1079 paths.
... Processed 200 out of 1079 paths.
... Processed 300 out of 1079 paths.
... Processed 400 out of 1079 paths.
... Processed 500 out of 1079 paths.
... Processed 600 out of 1079 paths.
... Processed 700 out of 1079 paths.
... Processed 800 out of 1079 paths.
```

Figure B.4: Status Form During Analysis



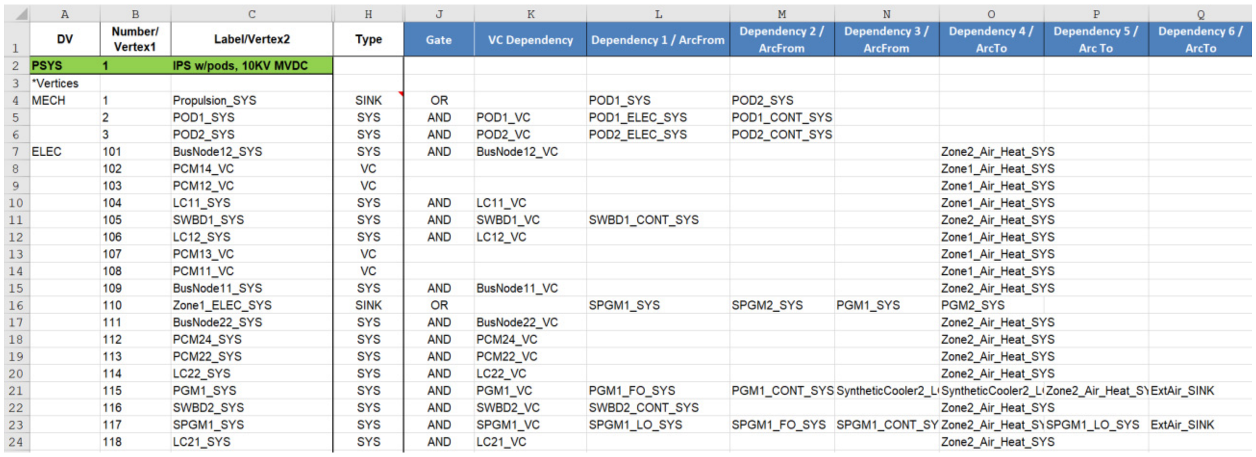
```
... Processed 600 out of 1079 paths.
... Processed 700 out of 1079 paths.
... Processed 800 out of 1079 paths.
... Processed 900 out of 1079 paths.
... Processed 1000 out of 1079 paths.
... Processed 1079 out of 1079 paths.
... Finished in 31 seconds.
... Completed Path Repair
Executing Fixes to Arc/Edge Connections...........
... Finished in 0 seconds.
... Completed Connection Repair
Updating Data Set............
... Completed Data Set Update
Adding Vertices to Workbook...........
... Finished in 8 seconds.
... Completed Adding Vertices
... Finished in 64 seconds.
... Completed Data Import!!!!!

857 resulting systems.
506 resulting vital components.
0 resulting unknown components.

Done. Close the form.
```

Figure B.5: Status Form Following Analysis Completion

B.2 Data Format

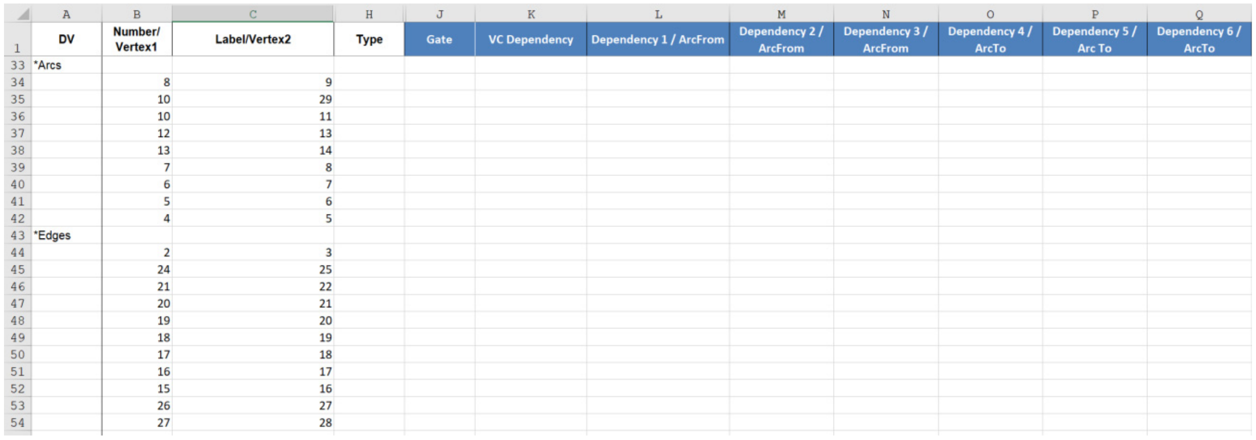
B.2.1 VT Ship Synthesis Model

Data formatting within the VT SSM takes the form shown in Figure B.6 and Figure B.7. With respect to the example in Figure B.6, cell A2 (top-left) contains the referenced system name ("PSYS") followed by the system option number (1) in cell B2. The system option number allows for multiple system descriptions to be listed sequentially on the same worksheet, with at least one empty row required between system options. Underneath the system name, tags "*Vertices", "*Arcs", and "*Edges" indicate the start of corresponding data sections to the deactivation analysis code. Under the "*Vertices" section, any following names (e.g. "MECH", "ELEC", "CW", etc.) indicate individual plex names for vertices defined next to and following the specified plex name row.



	A	B	C	H	J	K	L	M	N	O	P	Q
1	DV	Number/ Vertex1	Label/Vertex2	Type	Gate	VC Dependency	Dependency 1 / ArcFrom	Dependency 2 / ArcFrom	Dependency 3 / ArcFrom	Dependency 4 / ArcTo	Dependency 5 / Arc To	Dependency 6 / ArcTo
2	PSYS	1	IPS w/pods, 10KV MVDC									
3	*Vertices											
4	MECH	1	Propulsion_SYS	SINK	OR		POD1_SYS	POD2_SYS				
5		2	POD1_SYS	SYS	AND	POD1_VC	POD1_ELEC_SYS	POD1_CONT_SYS				
6		3	POD2_SYS	SYS	AND	POD2_VC	POD2_ELEC_SYS	POD2_CONT_SYS				
7	ELEC	101	BusNode12_SYS	SYS	AND	BusNode12_VC				Zone2_Air_Heat_SYS		
8		102	PCM14_VC	VC						Zone1_Air_Heat_SYS		
9		103	PCM12_VC	VC						Zone1_Air_Heat_SYS		
10		104	LC11_SYS	SYS	AND	LC11_VC				Zone1_Air_Heat_SYS		
11		105	SWBD1_SYS	SYS	AND	SWBD1_VC	SWBD1_CONT_SYS			Zone2_Air_Heat_SYS		
12		106	LC12_SYS	SYS	AND	LC12_VC				Zone1_Air_Heat_SYS		
13		107	PCM13_VC	VC						Zone1_Air_Heat_SYS		
14		108	PCM11_VC	VC						Zone1_Air_Heat_SYS		
15		109	BusNode11_SYS	SYS	AND	BusNode11_VC				Zone2_Air_Heat_SYS		
16		110	Zone1_ELEC_SYS	SINK	OR		SPGM1_SYS	SPGM2_SYS	PGM1_SYS	PGM2_SYS		
17		111	BusNode22_SYS	SYS	AND	BusNode22_VC				Zone2_Air_Heat_SYS		
18		112	PCM24_SYS	SYS	AND	PCM24_VC				Zone2_Air_Heat_SYS		
19		113	PCM22_SYS	SYS	AND	PCM22_VC				Zone2_Air_Heat_SYS		
20		114	LC22_SYS	SYS	AND	LC22_VC				Zone2_Air_Heat_SYS		
21		115	PGM1_SYS	SYS	AND	PGM1_VC	PGM1_FO_SYS	PGM1_CONT_SYS	SyntheticCooler2_L	SyntheticCooler2_L	Zone2_Air_Heat_S	ExtAir_SINK
22		116	SWBD2_SYS	SYS	AND	SWBD2_VC	SWBD2_CONT_SYS			Zone2_Air_Heat_SYS		
23		117	SPGM1_SYS	SYS	AND	SPGM1_VC	SPGM1_LO_SYS	SPGM1_FO_SYS	SPGM1_CONT_SY	Zone2_Air_Heat_S	SPGM1_LO_SYS	ExtAir_SINK
24		118	LC21_SYS	SYS	AND	LC21_VC				Zone2_Air_Heat_SYS		

Figure B.6: Sample VT SSM Input Vertex List



	A	B	C	H	J	K	L	M	N	O	P	Q
1	DV	Number/ Vertex1	Label/Vertex2	Type	Gate	VC Dependency	Dependency 1 / ArcFrom	Dependency 2 / ArcFrom	Dependency 3 / ArcFrom	Dependency 4 / ArcTo	Dependency 5 / Arc To	Dependency 6 / ArcTo
33	*Arcs											
34		8	9									
35		10	29									
36		10	11									
37		12	13									
38		13	14									
39		7	8									
40		6	7									
41		5	6									
42		4	5									
43	*Edges											
44		2	3									
45		24	25									
46		21	22									
47		20	21									
48		19	20									
49		18	19									
50		17	18									
51		16	17									
52		15	16									
53		26	27									
54		27	28									

Figure B.7: Sample VT SSM Connections List

In the "*Vertices" section, the data structure necessary for deactivation analysis follows these given rules according to vertex type/role:

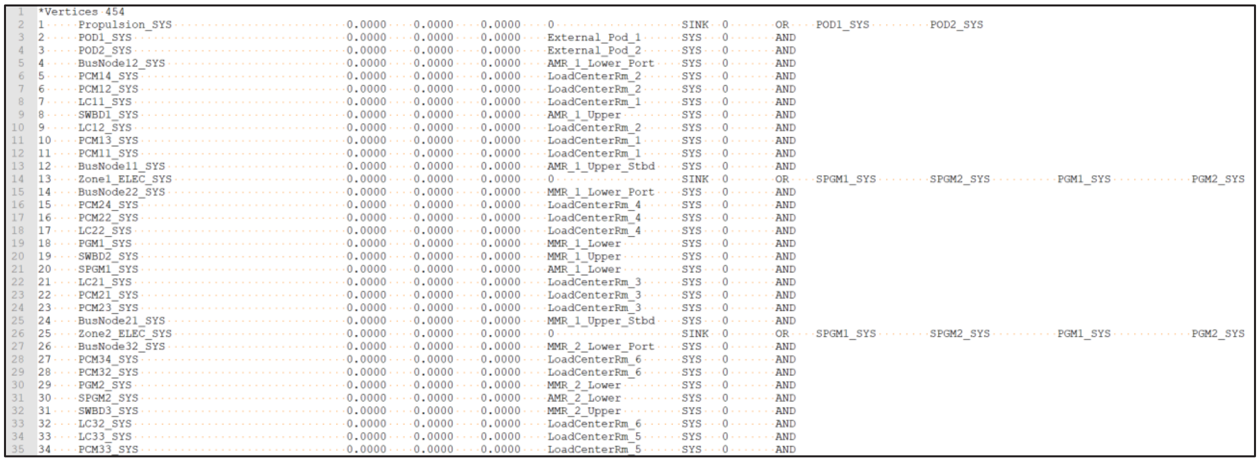
- Systems
    - Defined as a "SYS" in the H ("Type") column and having a name that is unique and ending in "_SYS".
    - Column B contains the vertex number for table identification and object tracking in the deactivation analysis. The vertex number must be unique.
    - Column K contains the identification of a primary associated VC for the system (if applicable) for an explicit connection from that VC.
    - Column J contains the gate-type for source connections to the given system. A system requires all connections to function with an "AND" gate and only 1 to function with an "OR" gate. Combinations of AND/OR should be split amongst multiple interconnected systems.
    - Columns L-N contain implicit source connections to the system from vertices outside the current plex. Explicit connections (within a plex) are listed in the "*Arcs" and "*Edges" sections.
    - Columns O-Q contain implicit sink connections from the system to vertices outside the current plex. Explicit connections (within a plex) are listed in the "*Arcs" and "*Edges" sections.
- Vital Components
    - Defined as a "VC" in the H ("Type") column and having a name that is unique and ending in "_VC".
    - Column B contains the vertex number for table identification and object tracking in the deactivation analysis. The vertex number must be unique.
    - Columns K-N are not typically used. If necessary, follow same rules as Systems.
    - Columns O-Q contain implicit sink connections from the system to vertices outside the current plex. Explicit connections (within a plex) are listed in the "*Arcs" and "*Edges" sections.
- SOURCES
    - Defined as a "SYS" or "SOURCE" in the H ("Type") column and having a name that is unique and ending in "_SYS" or "_SOURCE" ("_SYS" preferred). Explicit definition as a source is not required
    - Follows all other System rules.

- SINKS
  - Defined as a "SINK" or "LOOP" in the H ("Type") column and having a name that is unique and ending in "_SYS" or "_SINK" ("_SYS" preferred). "LOOP" signifies a closed-loop system and there should be no data in columns K-Q as the sink also acts as the source.
  - Column B contains the vertex number for table identification and object tracking in the deactivation analysis. The vertex number must be unique.
  - Column K is not typically used.
  - Column J contains the gate-type for source connections to the sink. A sink requires all connections to function with an "AND" gate and only 1 to function with an "OR" gate. Combinations of AND/OR should be split amongst multiple source systems.
  - Columns L-Q contain the explicit source dependencies for the sink. All sources must exist within the same plex, implicit connections to other systems is not supported.

The "*Arcs" and "*Edges" sections define connections of adjacent vertices using pairs of vertex numbers in columns B & C as shown in Figure B.7. For arcs, the vertex pairs are listed with the source first (source in column B).

B.2.2 Pajek ".NET" File

The Pajek system architecture file uses a ".net" file extension. Data contained within each file is stored in a space-delimited plain text format shown in Figure B.8. The standard Pajek data file has been modified in this thesis to store additional information and retain Pajek compatibility.



```
*Vertices 454
1    Propulsion_SYS        0.0000  0.0000  0.0000  0                  SINK  0   OR   POD1_SYS   POD2_SYS
2    POD1_SYS              0.0000  0.0000  0.0000  External_Pod_1     SYS   0   AND
3    POD2_SYS              0.0000  0.0000  0.0000  External_Pod_2     SYS   0   AND
4    BusNode12_SYS         0.0000  0.0000  0.0000  AMR_1_Lower_Port   SYS   0   AND
5    PCM14_SYS             0.0000  0.0000  0.0000  LoadCenterRm_2     SYS   0   AND
6    PCM12_SYS             0.0000  0.0000  0.0000  LoadCenterRm_2     SYS   0   AND
7    LC11_SYS              0.0000  0.0000  0.0000  LoadCenterRm_1     SYS   0   AND
8    SWBD1_SYS             0.0000  0.0000  0.0000  AMR_1_Upper        SYS   0   AND
9    LC12_SYS              0.0000  0.0000  0.0000  LoadCenterRm_2     SYS   0   AND
10   PCM13_SYS             0.0000  0.0000  0.0000  LoadCenterRm_1     SYS   0   AND
11   PCM11_SYS             0.0000  0.0000  0.0000  LoadCenterRm_1     SYS   0   AND
12   BusNode11_SYS         0.0000  0.0000  0.0000  AMR_1_Upper_Stbd   SYS   0   AND
13   Zone1_ELEC_SYS        0.0000  0.0000  0.0000  0                  SINK  0   OR   SPGM1_SYS  SPGM2_SYS  PGM1_SYS  PGM2_SYS
14   BusNode22_SYS         0.0000  0.0000  0.0000  MMR_1_Lower_Port   SYS   0   AND
15   PCM24_SYS             0.0000  0.0000  0.0000  LoadCenterRm_4     SYS   0   AND
16   PCM22_SYS             0.0000  0.0000  0.0000  LoadCenterRm_4     SYS   0   AND
17   LC22_SYS              0.0000  0.0000  0.0000  LoadCenterRm_4     SYS   0   AND
18   PGM1_SYS              0.0000  0.0000  0.0000  MMR_1_Lower        SYS   0   AND
19   SWBD2_SYS             0.0000  0.0000  0.0000  MMR_1_Upper        SYS   0   AND
20   SPGM1_SYS             0.0000  0.0000  0.0000  AMR_1_Lower        SYS   0   AND
21   LC21_SYS              0.0000  0.0000  0.0000  LoadCenterRm_3     SYS   0   AND
22   PCM21_SYS             0.0000  0.0000  0.0000  LoadCenterRm_3     SYS   0   AND
23   PCM23_SYS             0.0000  0.0000  0.0000  LoadCenterRm_3     SYS   0   AND
24   BusNode21_SYS         0.0000  0.0000  0.0000  MMR_1_Upper_Stbd   SYS   0   AND
25   Zone2_ELEC_SYS        0.0000  0.0000  0.0000  0                  SINK  0   OR   SPGM1_SYS  SPGM2_SYS  PGM1_SYS  PGM2_SYS
26   BusNode32_SYS         0.0000  0.0000  0.0000  MMR_2_Lower_Port   SYS   0   AND
27   PCM34_SYS             0.0000  0.0000  0.0000  LoadCenterRm_6     SYS   0   AND
28   PCM32_SYS             0.0000  0.0000  0.0000  LoadCenterRm_6     SYS   0   AND
29   PGM2_SYS              0.0000  0.0000  0.0000  MMR_2_Lower        SYS   0   AND
30   SPGM2_SYS             0.0000  0.0000  0.0000  AMR_2_Lower        SYS   0   AND
31   SWBD3_SYS             0.0000  0.0000  0.0000  MMR_2_Upper        SYS   0   AND
32   LC32_SYS              0.0000  0.0000  0.0000  LoadCenterRm_6     SYS   0   AND
33   LC33_SYS              0.0000  0.0000  0.0000  LoadCenterRm_5     SYS   0   AND
34   PCM33_SYS             0.0000  0.0000  0.0000  LoadCenterRm_5     SYS   0   AND
```

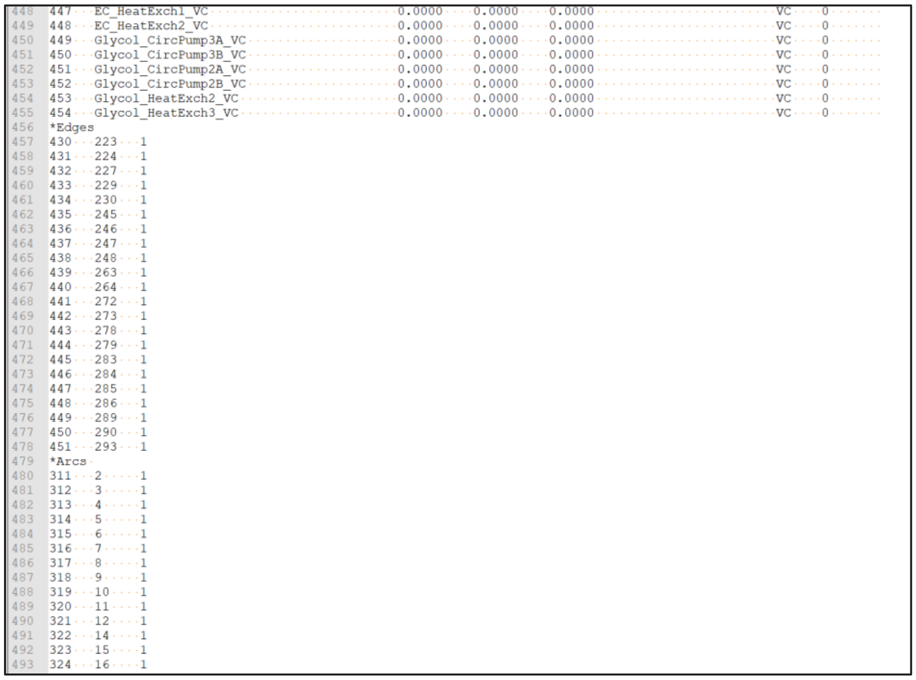
Figure B.8: Sample Pajek Vertex List

Although organized neatly in Figure B.8, data contained within the Pajek ".net" file format only requires space or tab delimiting between text to separate objects, so vertex names must not contain any spaces. This format was the basis for the VT SSM format and many similarities exist between the two.

The vertex definition section starts the Pajek file. The first line of text contains “*Vertices” and a count of the total number of vertices. From left to right starting on the next line, the delimited columns of data contain:

- Unique vertex number, counted incrementally starting at 1.
- Unique vertex name following convention set for VT SSM. Vertex name endings are only applicable to VT work including deactivation analysis and do not matter to Pajek.
- X, Y, and Z coordinates for the 3-D spatial position of vertices for Pajek visualization.
- Compartment name for each vertex.
- Vertex type, specified as either “VC”, “SYS”, or “SINK” as appropriate.
- Placeholder for additional information, such as electrical requirement. Current default value is 0.
- Gate-type for source connections to a sink. If object is a sink in a closed-loop system, the value is “LOOP”.
- Delimited list of explicit sources (same plex) for a sink or implicit external sources (different plex) for a system not acting as a sink.

Arcs and edges in the Pajek file are defined similarly to the VT SSM as shown in Figure B.9, with each connection pair listed under the appropriate section heading (“*Arcs” or “*Edges”) and each arc source preceding the sink. Pajek allows a third column containing connection weighting data, which is currently not utilized in the Deactivation Analysis Tool and is set to a default value of 1.



```
447   EC_HeatExch1_VC                     0.0000    0.0000    0.0000                        VC    0
448   EC_HeatExch2_VC                     0.0000    0.0000    0.0000                        VC    0
449   Glycol_CircPump3A_VC                0.0000    0.0000    0.0000                        VC    0
450   Glycol_CircPump3B_VC                0.0000    0.0000    0.0000                        VC    0
451   Glycol_CircPump2A_VC                0.0000    0.0000    0.0000                        VC    0
452   Glycol_CircPump2B_VC                0.0000    0.0000    0.0000                        VC    0
453   Glycol_HeatExch2_VC                 0.0000    0.0000    0.0000                        VC    0
454   Glycol_HeatExch3_VC                 0.0000    0.0000    0.0000                        VC    0
*Edges
430   223   1
431   224   1
432   227   1
433   229   1
434   230   1
435   245   1
436   246   1
437   247   1
438   248   1
439   263   1
440   264   1
441   272   1
442   273   1
443   278   1
444   279   1
445   283   1
446   284   1
447   285   1
448   286   1
449   289   1
450   290   1
451   293   1
*Arcs
311   2     1
312   3     1
313   4     1
314   5     1
315   6     1
316   7     1
317   8     1
318   9     1
319   10    1
320   11    1
321   12    1
322   14    1
323   15    1
324   16    1
```

Figure B.9: Sample Pajek Connection List

B.3 Instructions

The steps necessary to operate the Deactivation Analysis Tool are as follows:

1- Identify and collate system data for use in the deactivation analysis tool using Microsoft Excel or Pajek-style ".net" files according to format guidance provided in Section B.2.

2- List data sources in the input fields.

   a. For Excel input, list the system (worksheet) name and option number. Optionally specify the plex filters to apply.

   b. For ".net" file input, specify a system name for the data file and the fully-qualified file path.

3- Set optional flags for export to a file or to Microsoft Visio after the analysis has completed. Export to Visio requires a current installation of Visio 2016+ on the user's computer.

4- Set flag for ignoring warning messages & errors. It is generally recommended to not ignore errors unless the input files have already been vetted and the deactivation analysis process is being automated. (Default: False)

5- Press the "Run Import" button to import the data and start the analysis.

6- Wait for final competition of analysis computations. If the process needs to be aborted, first try the "Esc" key to stop the analysis. If this fails, end the Excel process in Windows Task Manager.

Appendix C

LEAPS Network Translator Tool Operation

C.1 Disclaimer

The LEAPS Network Translator has been built against the U.S. Navy's Leading-Edge Architecture for Prototyping Systems (LEAPS) v5.1.2 data repository and targeting systems developed in the Smart Ship System Design (S3D) v2.1 tool suite. Both projects are currently under development and future revisions may break current compatibility. Due to export restrictions, access to this tool might not be currently available.

C.2 Instructions

System databases used in the LEAPS Network Translator are expected to be developed by the S3D Schematic Designer. Object properties assigned to system components by this tool are necessary for accurate flow and system architecture analysis.

Using a system database containing one or more valid concepts,

1- Launch the "System Builder" application containing the LEAPS Network Translator tool code.

2- From the "File" menu, select "Open Ship Database" to load the LEAPS/S3D database.

   a. Specify fully-qualified folder path to the database source folder and the name of the database (same as containing folder). Select "OK".

   b. Choose the appropriate ship study from the list of available studies. Select "OK".

   c. Choose the appropriate ship concept from the list of available concepts. Select "OK".

3- From the "Admin" menu, diagram the system flow direction by selecting "Set Flow Direction"-> "Ship".

   a. May be disregarded if previously run.

4- System export options are available in the "Trace Diagram" menu under "Pajek File"-> "Ship". Select "Directed" for a directed arc graph or "Undirected" for an undirected arc (edge) graph.

   a. Choose the appropriate ship concept from the list of available concepts. Select "OK".

   b. Choose the appropriate diagram from the list of available diagrams. Select "OK".

   c. Find Pajek-style data output in the software dialog box and saved to the text file "[Database Name].net" in the application execution directory.