SYSTEMS ENGINEERING
Research Center

Development and Application of FACT Portfolio Management Capability

Technical Report SERC-2016-TR-112

September 12, 2016

Principal Investigator:

Daniel C. Browne, Georgia Institute of Technology

Research Team:

D. David Fullmer, Jr.

Todd Shayler

Sponsor: Marine Corps Systems Command – Log IT War Room

The Systems Engineering Research Center (SERC) is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Office of the Assistant Secretary of Defense for Research and Engineering (ASD(R&E)) under Contract HQ0034-13-D-0004 (Task Order 0047).

TABLE OF CONTENTS

This page intentionally left blank

BACKGROUND AND OBJECTIVES

The Systems Engineering Research Center (SERC) and Georgia Tech Research Institute (GTRI) have participated in previous research efforts supporting the Marine Corps in the analysis and management of portfolios of systems and software solutions. In this research effort the SERC and GTRI expanded on that previous work to begin formalizing portfolio management methods into a reusable framework and explore new methods and strategies and that are extendable to systems of systems. Specifically, the research effort is comprised of three tasks:

1. Building upon work from RT-112, new modeling approaches and methodologies need to be developed and applied to the Log IT War Room analysis. For RT-112, the focus was on the Global Combat Support System – Marine Corps (GCSS-MC) which is one system of many with the Log IT portfolio. For that challenge, portfolio management strategies were applied to evaluate how different components could be combined to offer a complete solution for GCSS-MC. In this challenge, the strategies applied for GCSS-MC need to be reviewed for how they scale to a more complex system of systems and modified as necessary. Additionally, evaluation and scoring methods need to be developed which are specifically geared towards software-heavy systems such as those within the Log IT portfolio. Possible software attributes to be considered as inputs to these evaluation methods should be lines of code, open systems architecture, code quality, modularity, and age.

2. An initial task of the greater effort will require a complete functional decomposition of the tools and processes within the Log IT portfolio. In order to accomplish this effectively, a proper process, data collection, and organization strategy need to be defined and implemented for the team to operate. Proper execution of this task will ease the transition from functional decomposition to portfolio re-aggregation and evaluation. The research goal in this task is to identify the proper data elements that need to be collected for items within the Log IT portfolio so that modeling techniques can be applied. This will include developing a technique utilizing these data element inputs to identify overlaps and gaps in capabilities.

3. Data visualization techniques need to be designed and implemented to offer insightful visualizations to decision makers. Visualizations will be required for inspecting the functional decomposition as well as the scoring methodology for the portfolio re-aggregation. Preferred implementation of these visualizations is within the same tool supporting the functional decomposition and data gathering.

OVERVIEW AND MOTIVATION

Management of a portfolio of capabilities, generally realized in a suite of physical systems, is a common problem faced by the agencies and programs within the Department of Defense (DoD). Standardized methods of tackling these problems have been developed over time, with a focus on sound Systems Engineering (SE) principles and tools. Although a standard process is commonly executed in solving these portfolio management problems, a standardized suite of tools is non-existent to support that process. Most often, new tools are developed by each SE team, sometimes leveraging a past tool utilized by the team, but often time not being reused for future efforts. SERC RT-112 began the effort of formalizing some of these methods into tools [1].

This limited reuse of tools is a primary motivator to the SERC for this research: to capture the standard SE process and develop a toolset which allows analysis teams and Systems Engineers to focus time and effort on the analysis rather than management tools and processes. In order to inform the prioritization and development of capabilities, SERC worked closely with an ongoing project team: Log IT War Room. SERC remained engaged in the Log IT War Room problem while in parallel developing enhancements to the Portfolio Management and Analysis Toolset (PMAT) within Jupyter Notebooks [2].

The remainder of this report is organized as follows. First, SERC documents the standardized SE process applied to portfolio management problems. This was first documented in the SERC RT-112 final report [1] and has been updated based on this research. Following, the extensions to the PMAT capabilities are described through narrative and example graphics. Finally, future enhancements to the PMAT capabilities required to support time-based portfolio management challenges are described.

SYSTEMS ENGINEERING PROCESS FOR PORTFOLIO MANAGEMENT PROBLEMS

Portfolio management problems are well addressed when following a solid set of systems engineering principles and practices. As such, it is important that a systems engineer works closely with the customer as a facilitator. The overall process is described below to provide context for the discussion of the toolset that follows in the next section.

Define the Problem and List Alternatives

The process begins with a careful consideration of the problem. This entails both developing the problem statement (high-level goals) and compiling a list of alternatives that serve as potential solutions. Generally, that list of alternatives is captured within a Market Research Database (MRDB) that captures both currently available off-the-shelf solutions as well as potential low TRL (Technology Readiness Level) future solutions. Note that these are created iteratively, not sequentially. Decomposition and revision of the problem statement may highlight capability gaps in the alternatives and necessitate further consideration. Alternatives may also become irrelevant as the problem statement is refined.

Similarly, the inclusion (and exclusion) of options from the list of alternatives may lead to reconsideration of the problem statement. The goal is to make sure that all tacit needs are explicitly stated in the high-level goals. *This option seems relevant; why is it excluded? How do its capabilities differ from the problem as stated? Why are similar options included in the list of alternatives?*

The provenance of both the problem statement and the list of alternatives must be traced to ensure that they are accurate, complete, and current. It is possible that the customer may already have an idea of the high-level goals of the project and potential alternatives. If not, the facilitator is prepared to start from scratch, potentially organizing a workshop or developing a market survey. When developing the list of alternatives, it is important to forecast future alternatives relative to the problem scope, accounting for options that may not be available now, but will be by any implementation deadlines. Portfolio management exercises are often implemented in the not-so-near future.

Develop the Evaluation Framework

How are alternatives to be evaluated against the high-level goals of the project? The development of the evaluation framework requires extensive interaction with a small group of subject matter experts who possess a detailed understanding of the problem.

The high-level goals must be iteratively decomposed until they can be mapped to tangible alternatives. For example, a Log IT scenario (e.g. high demand for bandwidth, high latency, consistence technology solutions across programs) might be decomposed into a series of activities (e.g. make supply request, update inventory). These activities might then be mapped directly to the capabilities of the alternatives. This is an iterative process; there should be just enough levels to make the mappings traceable. Furthermore, the number of items in each level should increase

monotonically. That is, a 10-item level should not map to a 5-item level and then up to a 15-item level. This is indicative of information loss.

Next, define the mapping between each level. The map is a general function, but in most cases it takes the form of a 2-dimensional matrix with its rows corresponding to the domain level and its columns corresponding to the range level. The matrix will be populated with values from a scale defined by the facilitator. The scale may be numeric (e.g. 0, 1, 3, 9) or semantic (e.g. None, Little, Some, Much), though a semantic scale generally has an underlying numeric equivalence. Use clear, precise language. Avoid detailed scales with too much granularity; effort and time may be wasted in trying to differentiate between nearly equivalent values. Moreover, it may be necessary to hide the numeric meaning of the scale from the experts filling out the matrix. This information may affect their ratings either consciously or unconsciously.

Complete the Framework

The previous step developed the general framework for evaluating the alternatives against the high-level goals of the project. Now experts fill in the details (e.g. populate the matrices that map between levels). This may be accomplished with a survey or a workshop.

The usual biases that affect survey data apply here as well. It is important to make sure that responses are not influenced by superiors and that the loudest voice does not carry the day unquestioned. If possible, assess the accuracy of the data and explore any patterns.

If a workshop is necessary, be efficient. Distribute individual surveys at least 2 weeks before the workshop. Compile the data and identify disagreements for further discussion. Use anonymous voting methods to minimize biases and intimidation tactics. Furthermore, work with a respected mediator capable of controlling the experts, keeping the discussion moving, and minimizing unproductive conflict.

Pay attention to the proceedings. The validity of the framework may be assessed from the rate of progress. For example, slow progress may indicate that the map is too difficult to complete and further decomposition of the levels is necessary. Alternatively, rapid progress may indicate that the map is too obvious and there are too many levels. The framework may very well require further consideration.

It is not required that subject matter experts complete the framework. These experts provide insight into how alternatives function in the real world in their specific area of expertise, which may not cover the whole framework. The theoretical functionality may be studied with a quantitative model (e.g. physics models, battlefield simulation). The important result is a complete evaluation framework.

Test the Framework

Now, test the framework with various inputs and examine the results. If the analysis team is surprised by the results, determine the reason. The framework (or its data) may require further iterations. If the framework does not behave as desired, it is important to fix the problem at its source and not just artificially manipulate the results.

COMBINATORIAL DESIGN SPACE ANALYSIS TOOL

Once both the currently established components and the new candidate components for the Log IT system are identified and mapped to their associated functions it is possible to explore and analyze the combinatorial space in search for the right fit for the needs of the program. The “right” fit is determined based on a balance of capabilities,

cost, and risk. Although the solution space is often unintuitive, proper visualization techniques can aid understanding and facilitate analytical reasoning.

Visual Analytics

Visual analytics is the use of interactive visualizations to enable or advance the synthesis of information and derive insight from complex data [3]. For example, in 1869 Charles Minard a retired civil engineer produced the following information graphic in order to tell the story of Napoleon's Russian Campaign of 1812 (see Figure 1).

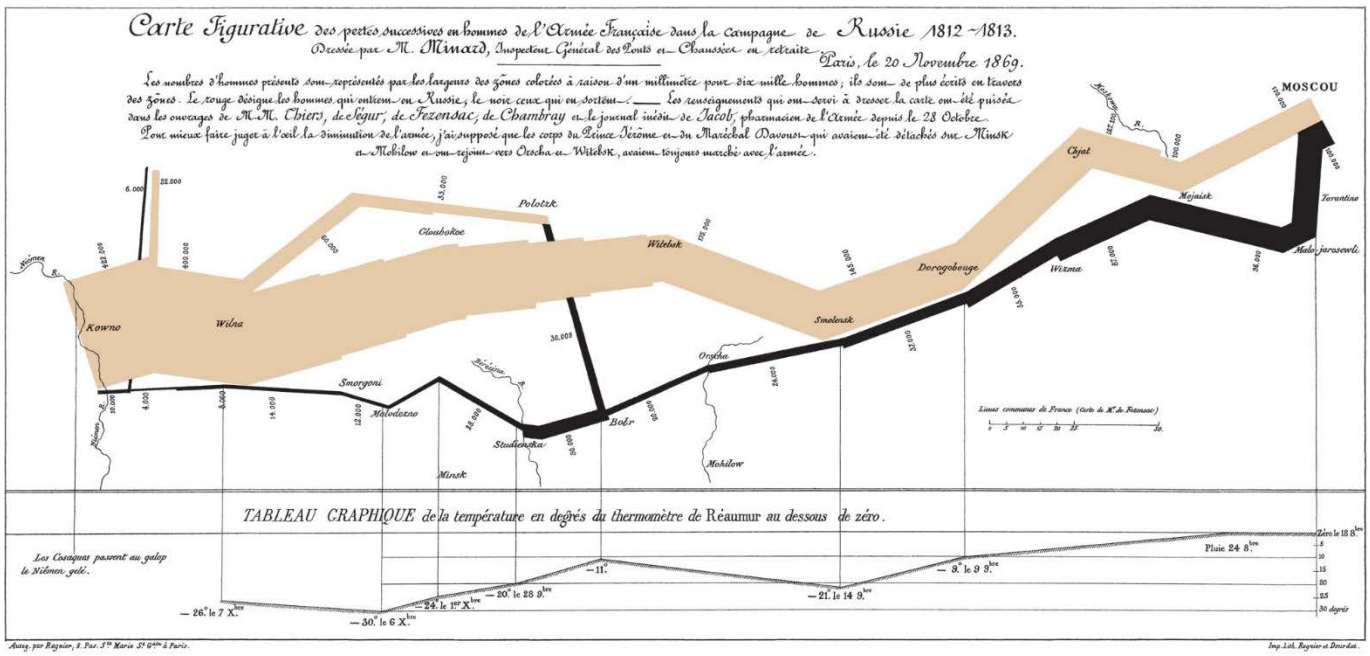

Figure 1. Napoleon's Russian Campaign of 1812 by Charles Minard

Starting from the left in the figure above the tan colored band depicts the size and location of Napoleon's army as Napoleon commenced his invasion of Russia. As the story unfolds Napoleon continues to rapidly push further into Russia in hopes of forcing the Russian army to battle. However, in each engagement the Russians retreat leaving the villages and crops burning, denying the French army the option of living off of the land. Without adequate food and supplies the French army dwindles as they continue to march deeper into Russia. The black band starting from the right and moving toward the left depicts the return trek of the French army and the continued suffering and losses.

The information graphic in this example is an effective storytelling visualization and it is reasonable that a similar set of visualizations are able to tell the design and engineering story for the combinatorial design process. However, this narrative will require specific characteristics:

- It should provide insight into the interconnectivity (the relationships) of the components, functions, and capabilities;
- It should provide decision support;
- It should provide traceability through design decisions and decision impacts;
- It should provide a clear description of the final solution or set of candidate solutions.

RELATIONSHIPS

The first challenge in a narrative is to introduce the elements of the story in such a way that all of the relationships between the elements are understood. The fundamental elements in the combinatorial design process are components, functions, and capabilities as illustrated below in Figure 2.

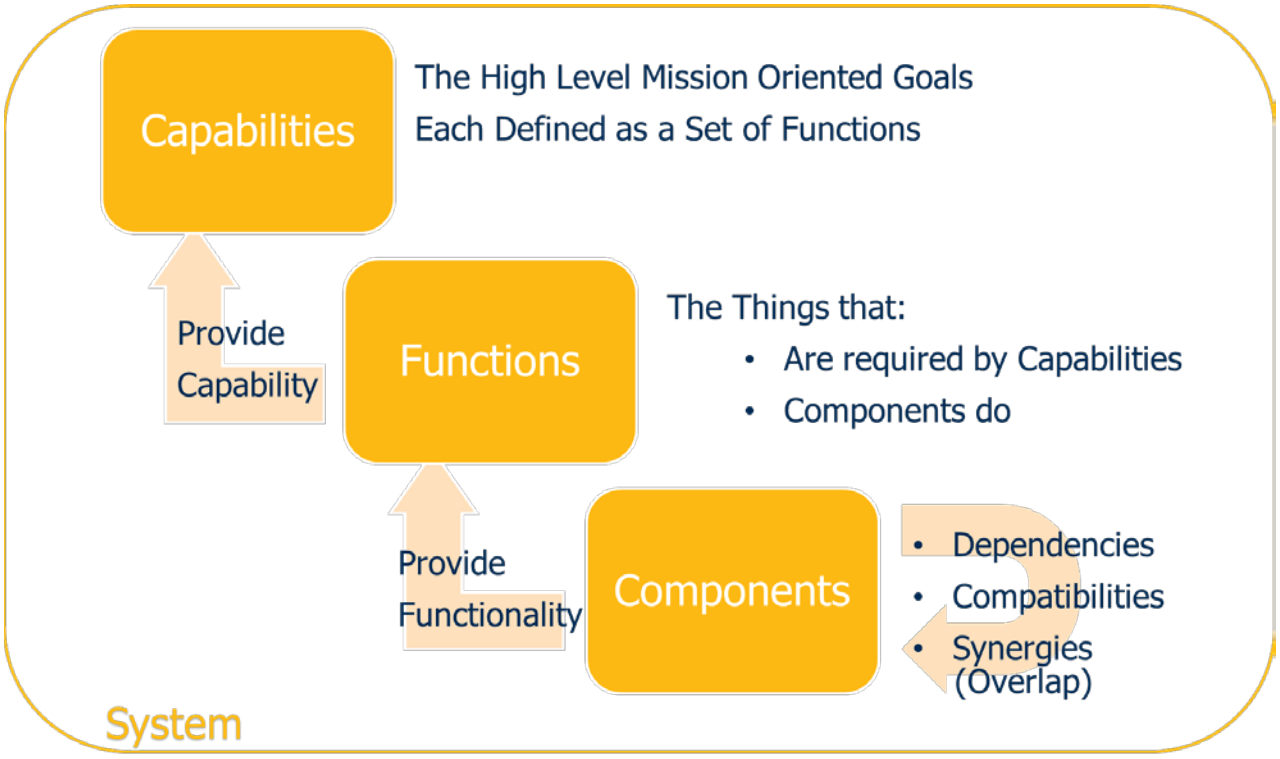

Figure 2. Types of Elements

Components are the parts of a system that provide functionality. For example, a component could be a radio or an engine. Components also have attributes associated with them, such as mass or cost. Functionality is captured as a function which is a specific process, action, or task that is performed. Capabilities represent a set of functions working together to meet needs. A system is an engineering solution to a set of needs. Systems are made up of interconnected components which work together to provide capabilities. System attributes are derived from the aggregation of its component's attributes. A system architecture is a class of systems that utilize a specific set of components to achieve a set of capabilities.

A problem formulation (i.e. Log IT) will have a specific set of needs, capabilities, functions, and components captured in machine- and human-friendly form (most often Excel). Checks are put in place to audit the data to help maintain integrity in the information being captured. The resulting information will resemble a network graph with different types of nodes. While there are many layouts that could be used for drawing a graph it makes sense to utilize the position of the nodes with some intrinsic meaning in order to emphasize the relationships within the types of nodes and between the different types of nodes.

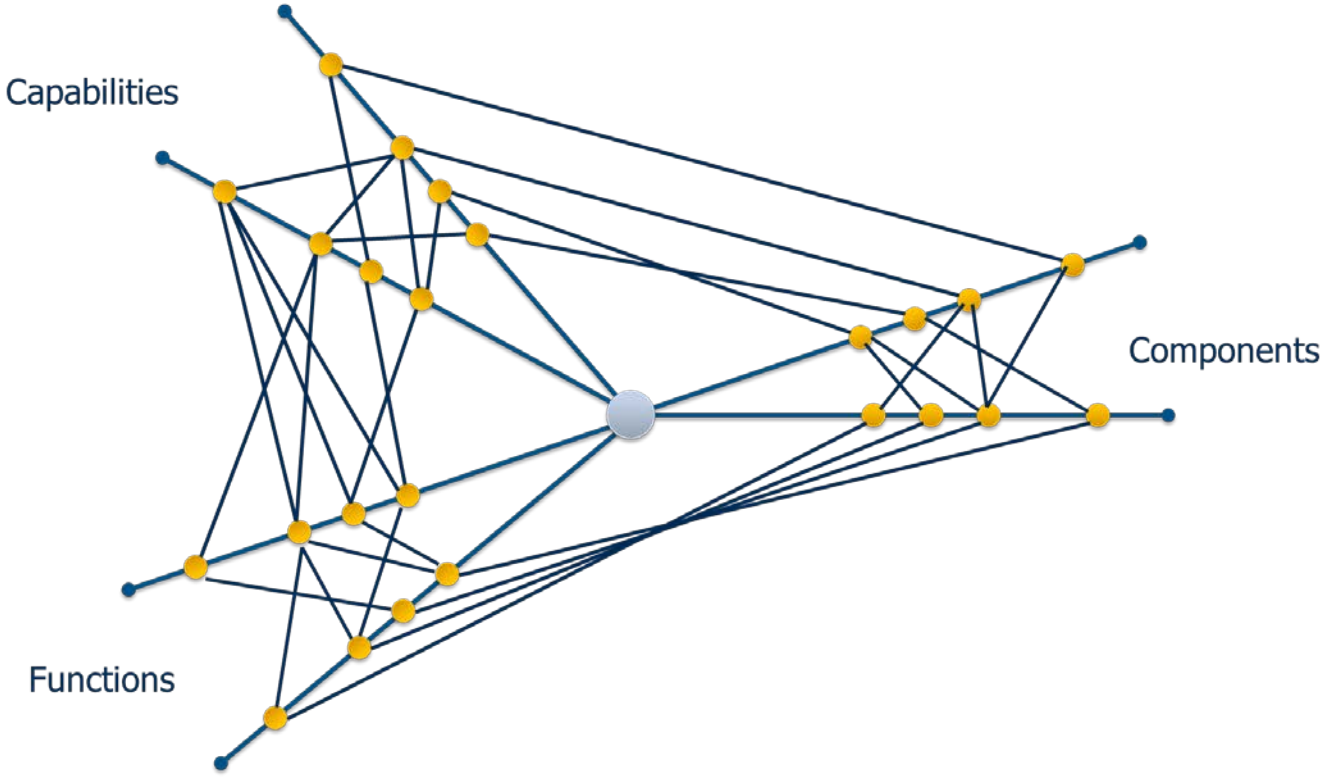

Figure 3. Hive Plot

Figure 3 above is an illustration of a Hive plot which groups the nodes by type and lays them out on a set of radial axes. The example above shows how capabilities, functions, and components are depicted. Each type of node has two axes assigned in order to allow the easy visualization of the relationships between the same types of node. For example it might be common for a set of functions to be packaged together in a component. This relationship would be apparent between the two axes of the same type. On the other hand, the components that provide a specific set of functions are shown between the axis of the components and the axis of the functions.

Notional Solution

The complete solution not only includes a representation of the finial state, but it also includes each intermediate state in a clear plan of progression. As the duration of a project increases the likelihood of the solution changing also increases. Particularly, where relatively fast evolving technology is involved.

A true solution for such a project must embrace the dual reality and needs to map out a solution from begin to end and enable the consideration of alternatives throughout its duration. Figure 4 below is an illustration of how the process might be visualized.

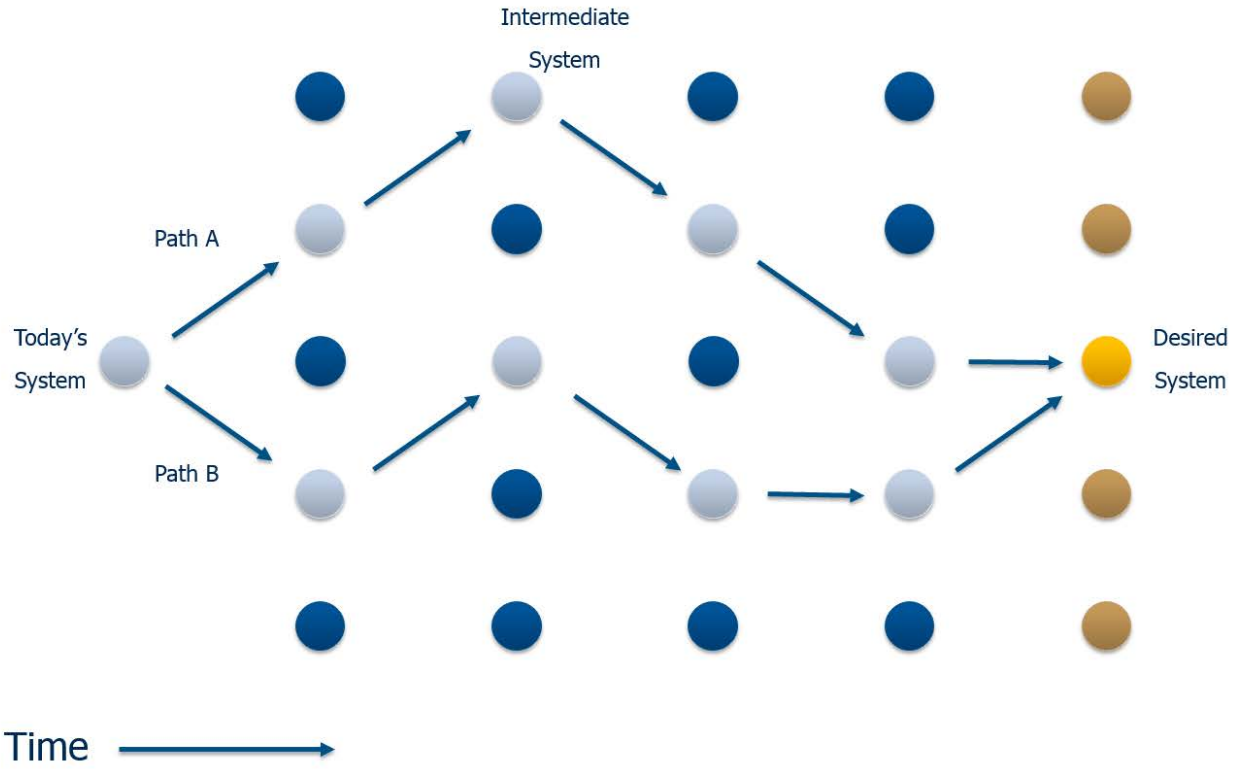

Figure 4. Portfolio Design and Change Paths over Time

The light gold point represents the currently desired system based on information collected from stakeholders. The darker gold points represent uncertainty in that desired system. There is recognition that there is some uncertainty such as in the understanding of the requirements or that the need may change or evolve over time. The light and dark blue points, specifically those along Paths A and B, represent intermediate states of the portfolio. Because of the processes involved in DoD acquisition or limitations from funding cycles, it is often the case that a portfolio cannot be transformed in a single time step, but rather requires incremental modifications where each intermediate state is offering the base set of required capability while also moving towards the desired state. There are many intermediate states which could be implemented; the light blue points represent those that fall along candidate paths of transition.

Implementation

The implementation is a framework designed for solving combinatorial design problems using a graph representation of the relationships between components, functions, and capabilities. The framework is developed in the Python programming language using a Jupyter Notebook [2]. The notebook paradigm provides a convenient development and rapid prototyping environment with access to both the strong analytical capabilities of the programming language and a strong integration of a web-based visualization capability. The Notebook is an effective user interface allowing the developed software to be packaged as a Python module for easy portability and extensibility toward a broad range of problems.

Often at the system level, engineering problems can be represented as a graph of multiple types of nodes and edges. Where the nodes represent various parts (whether physical or abstract) of the system and the edges represent the relationships between the parts. While these graphs can be difficult to visualize and analyze for decision making purposes, graphs are an effective representation for both computer and human consumption. The information the graphs are based on are the products of the research and experiences of subject matter experts.

In the example graph in Figure 5 below, the different types of nodes (component, function, and capability nodes) are color-coded and arranged in a hive chart as follows:

- Blue and orange nodes represent components;
- Green nodes represent functions;
- Purple nodes represent capabilities;
- Red nodes represent the various sets of conditions for which capabilities can be obtained;
- The edges between the orange and green nodes represent the relationship between components and functions, meaning a component offers a function;
- The edges between the green and red nodes represent the relationships between functions and the sets of conditions for obtaining capabilities;
- The edges between the red and purple nodes represent the relationships between the sets of conditions and the actual capabilities;
- The edges between the purple and blue nodes represent the cases where a capability can be directly obtained through the use of a single component.

The unused space between the orange nodes and blue nodes is a placeholder for capturing dependency and compatibility information between components.

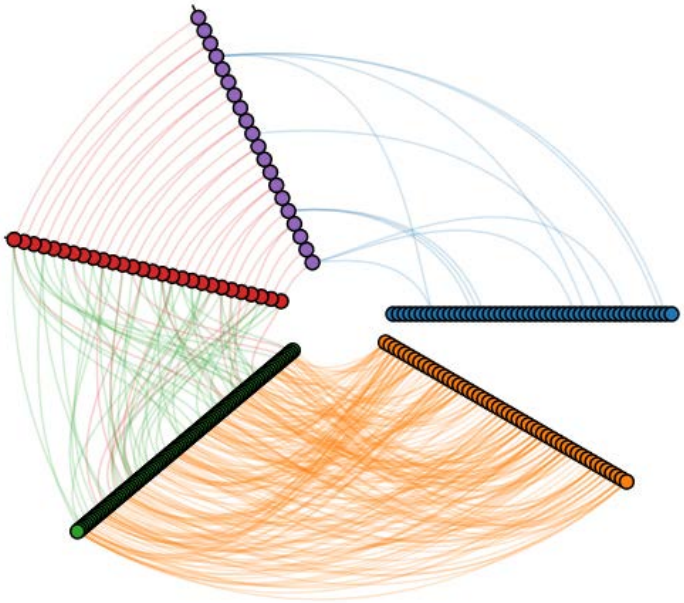

Figure 5. Example Hive Plot

The hive chart technique is an effective way of visually organizing the nodes in the graph in order to emphasize the types of relationships that exist between the different types of nodes. In addition, the hive technique coupled with interactive animations makes it a simple task to trace which combination of components produces which capabilities. However, the number of possible combinations is intractable for a typical number of components. Assuming that the system must have at least one component, there are 2^{50}-1=1,125,899,906,842,623 unique systems with 50 components.

The data used in this example are notional, consisting of generic components, functions, and capabilities. The framework is written based on the assumption that all of the data are real and are accessible from an appropriate authoritative source (i.e. Excel sheets, data files, or a database). However, given the nature of research and the availability of data, it is advantageous to include support for generating generic placeholders. A few of the features of the generic data generator are as follows:

- Data are available to support the development of the framework;
- Data characteristics may be changed to design a more robust framework;
- Real and generic data may be mixed to support a partial MRDB during project execution.

Given the number of possible unique systems and the nonlinear nature of the domain space a search algorithm (random combination generator, genetic algorithm, simulated annealing, etc.) cannot offer any guarantees as to the optimality of its results. However, rather than searching with the goal of finding the proverbial needle in the haystack, this approach advocates searching for the purpose of learning and characterizing patterns with respect to design decisions associated with the components and component interactions.

Consider for a moment which sample points are the most valuable in terms of providing the information needed to answer the following questions:

- Which components complement each other?
- How close in terms of cost and lead-time is the next capability?
- Which components or combinations of components provide the most value?

The information required to answer these questions hinge on an ability to value a solution. Once value can be calculated for a solution then complementing components, value to cost, and overall utility can be appreciated. In this approach value is assigned to each capability indicating the relative importance of each capability with respect is to each other (see Figure 6). In addition, solution-wide audit items are defined to capturing value toward workers such as overall solution complexity for overall solution cost.

Sample Custom Audit Functions

Example audit functions

```
def check_complexity(system):
    return +1

def check_budget(system):
    return +1

def check_operational_resources(system):
    return +1

audit_functions = [check_mass_margin, check_budget, check_power]
```

Figure 6. Example Audit Functions

An audit item can be based on any characteristic of an overall solution or any of the attributes of its components. For example, each component has a cost associated with it, so an audit item could look at each on the cost and compare the total cost with a set budget. The function defined below in Figure 7 illustrates one of the actual audit items for this example problem. The audit looks at the number of components employed within a solution and docks solutions with more than three components in an effort to drive solution complexity down.

```
def check_complexity(system):
    '''
        This audit function will take 50 points for each component over three
    '''
    s = sum(system)
    if s > 3:
        return -50*abs(s-3)
    else:
        return 0

m.ga_audit_functions = [check_complexity]
```

Figure 7. Complexity Audit Function

A sample population of the 1,125,899,906,842,623 unique possible systems is necessary to gain insight useful for decision-making. The value calculations act to enable a smart selection of the total possible systems to be the sample population. In this example, a modified genetic algorithm is utilized to search the options to produce a sample of 1000 good solutions. Since the goal is not to produce an optimal solution the ideal search results should show

- Good space exploration with visual variability
- Stable overall trends over multiple runs
- Quick run time

Figure 8 below is an example visualized resulting population. Each row is an individual solution portfolio and each column is a component. If the intersection of portfolio and a component is colored dark blue then that particular portfolio includes that particular component. At first look, a visual inspection of the density of the columns is an indication that some components are found to be more valuable than others to the search algorithm. When the algorithm is run multiple times this pattern should also be repeated as a condition for consideration as the sample populations for analysis.

```
m.run_ga(ngen=10, cxpb=1.0, mutpb=1.0, population_size=1000)
```

```
Min -469.0
Max 86.0
Avg -126.323
Std 71.67147738814933
```

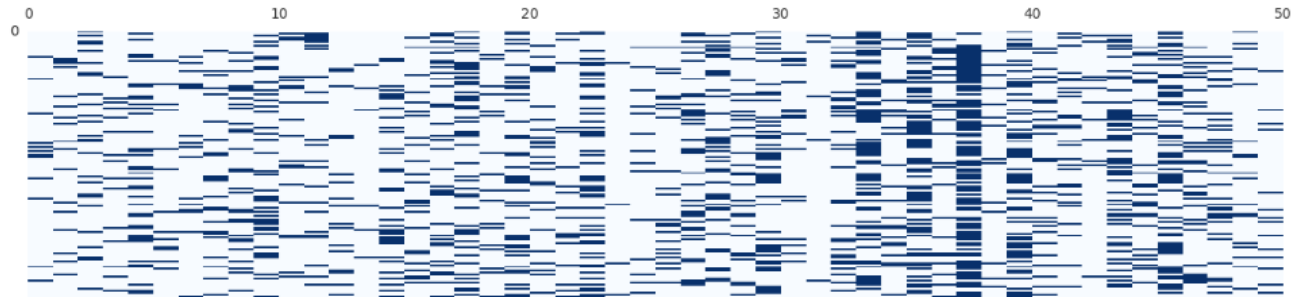

Figure 8. Population of a Genetic Algorithm

A slightly easier visualization of the column density is achieved though a simple bar chart as shown below in Figure 9. Here the x-axis corresponds to the components and the y-axis corresponds to the column density.

```
m.show_ga_cooccurrence(at_element=None)
```

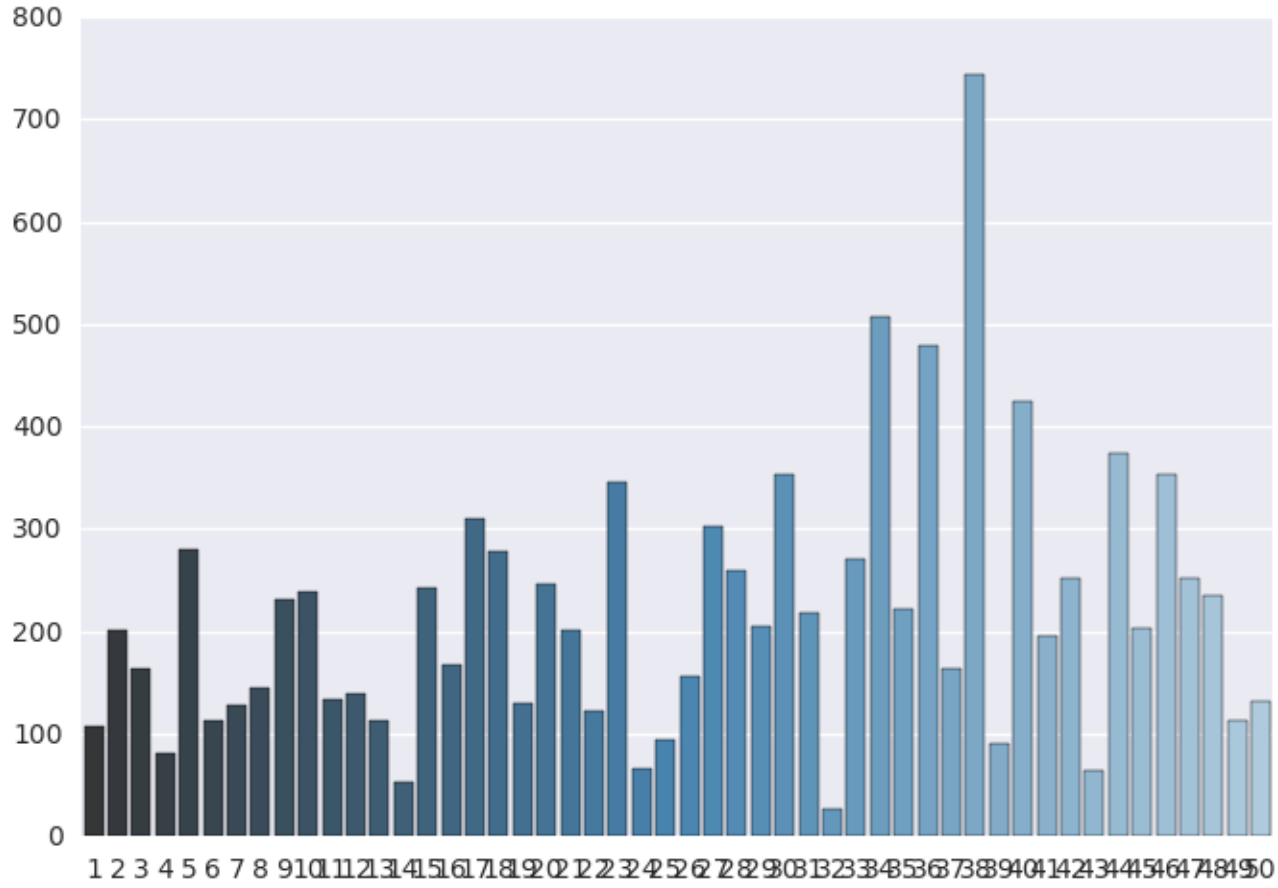

Figure 9. Population Component Frequency

At this point certain questions can be asked that may lead to some insight. For instance, the chart below shows the frequency at which components co-occur with Component 36 (see Figure 10).

```
m.show_ga_cooccurrence(at_element=36)
```

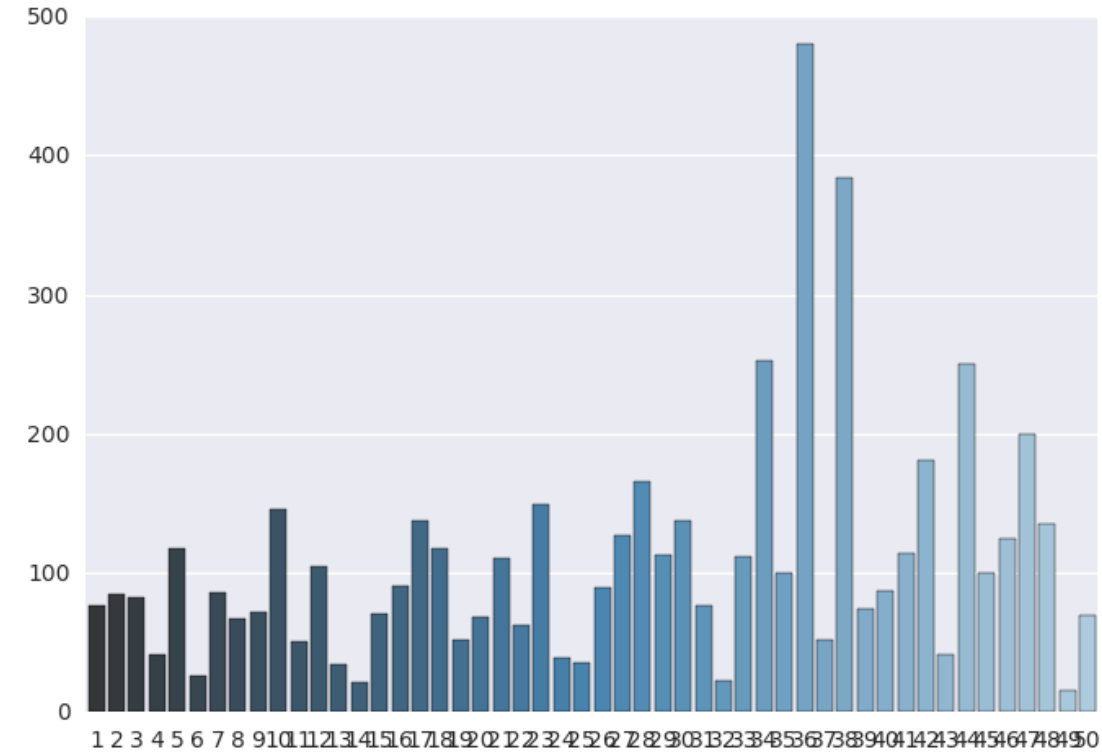

Figure 10. Population Co-Occurrence Frequency

Given that each of the individuals in the sample population are considered valuable, then it stands to reason that components that frequently co-occur with Component 36 are complementary with respect to value. This same information is visualized for all of the components simultaneously in the co-occurrence matrix in Figure 11. Each component is listed both on the x-axis and on the y-axis and the co-occurrence is indicated by the color of the intersections.

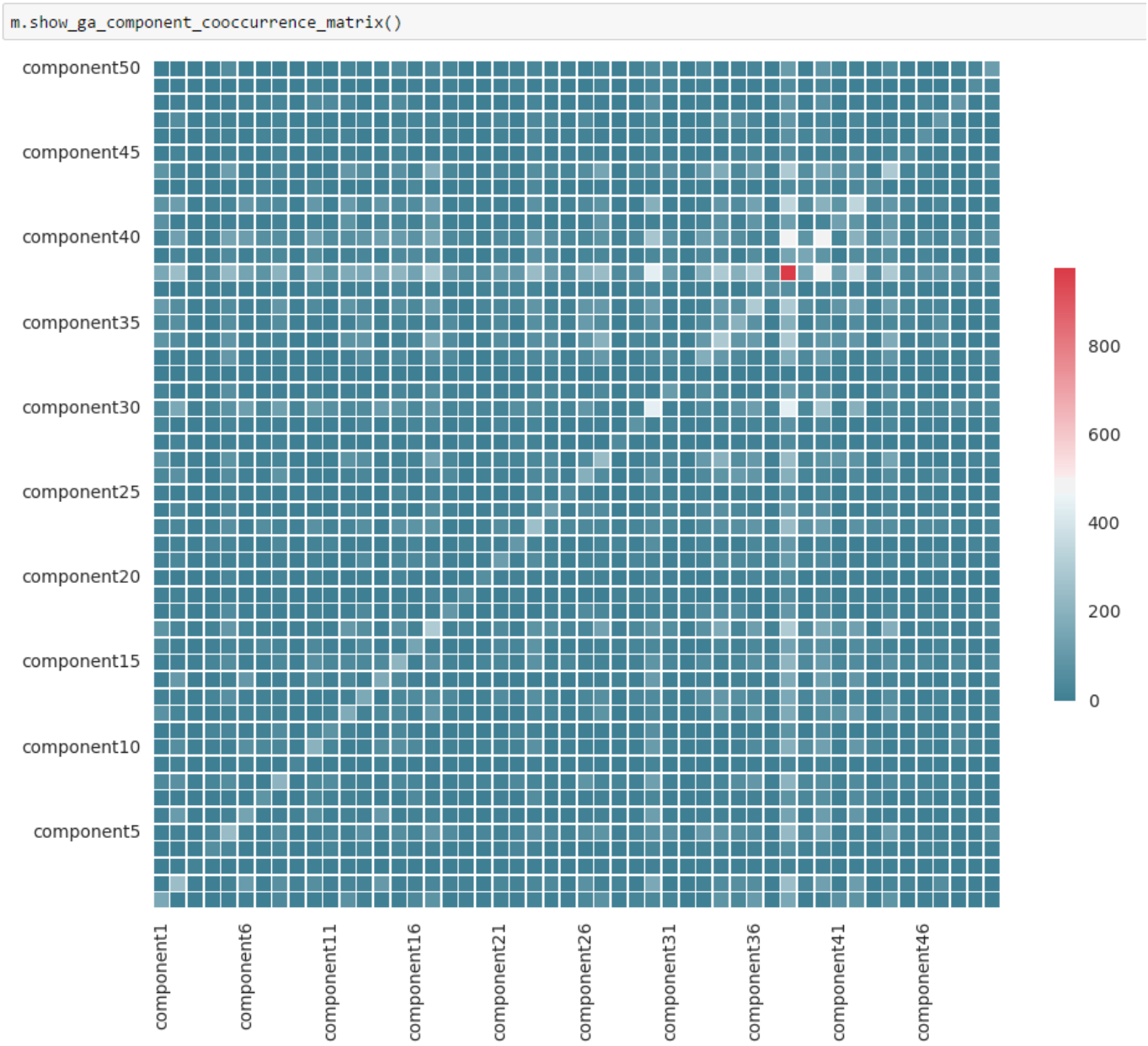

Figure 11. Population Co-Occurrence Matrix

The diagonal pattern in the matrix represents the same column density of the overall sample population show in the first bar chart. The intersections are color coded using a divergent color scale where dark blue to light blue represents co-occurrences of less than half of the sample populations. Light red to dark red represent co-occurrences of greater than half of the sample population. In this image, row 36 corresponds to the second bar chart shown earlier. In this view rows with lighter blues indicate components that couple well with other components. Conversely, rows with darker blues do not couple well with other components.

The co-occurrence matrix visualization can also be employed towards both functions and capabilities. However, both functions and capabilities are abstract concepts and realizable only through the inclusion of specific combinations of components. While the diagonals represent the column densities in terms of functions and capabilities respectively, the reasons for the variations in the densities might be complicated. For example, the dark blue intersections on the diagonal may be the result of low value be contributed to those functions or capabilities

by the stakeholder or could be due to an inherent difficulty in achieving the particular function or capability. Thus, the dark blues rows may indicate potential function gaps or capability gaps.

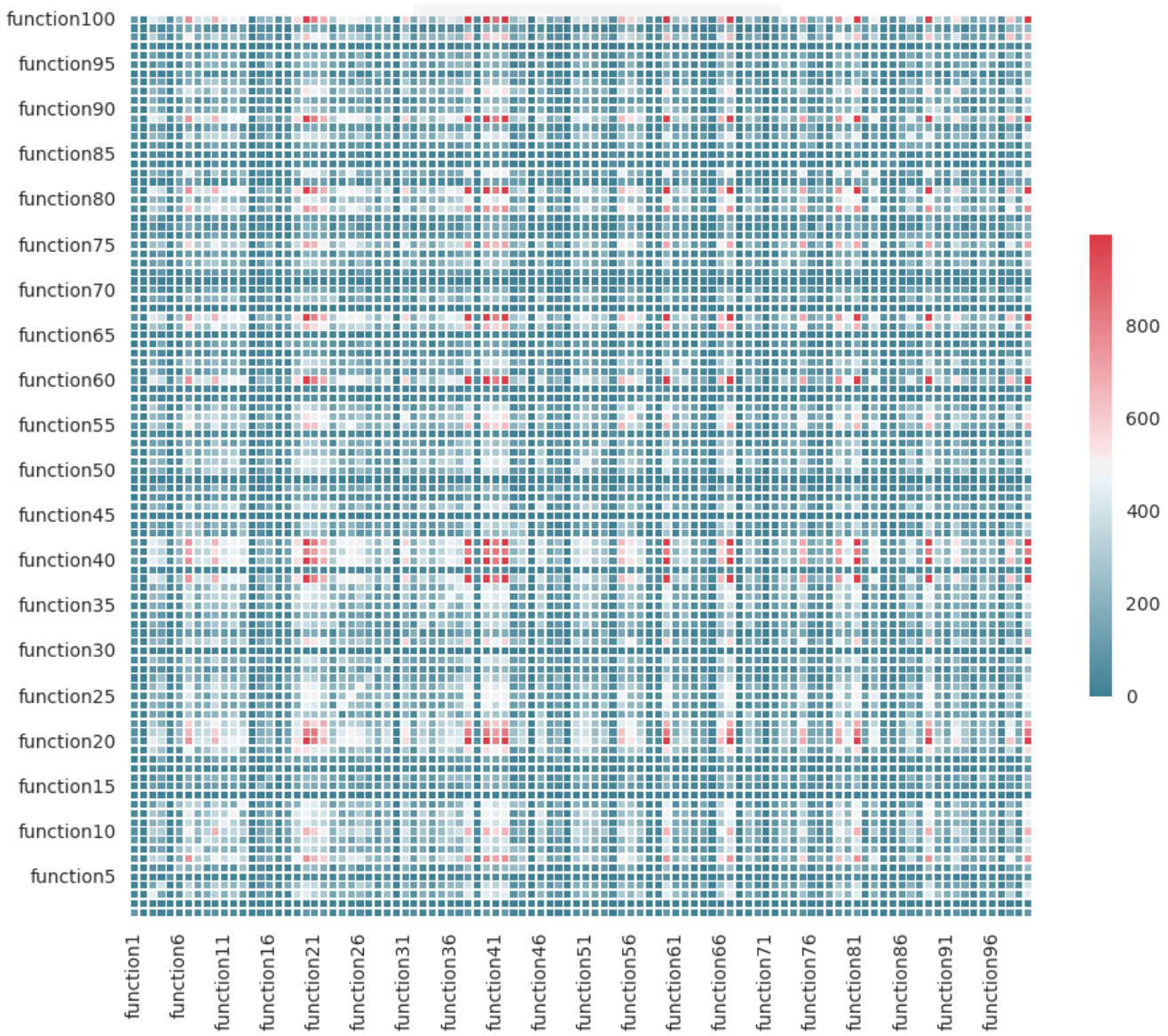

Figure 12. Function Co-Occurrence

```
m.show_ga_capability_cooccurrence_matrix()
```

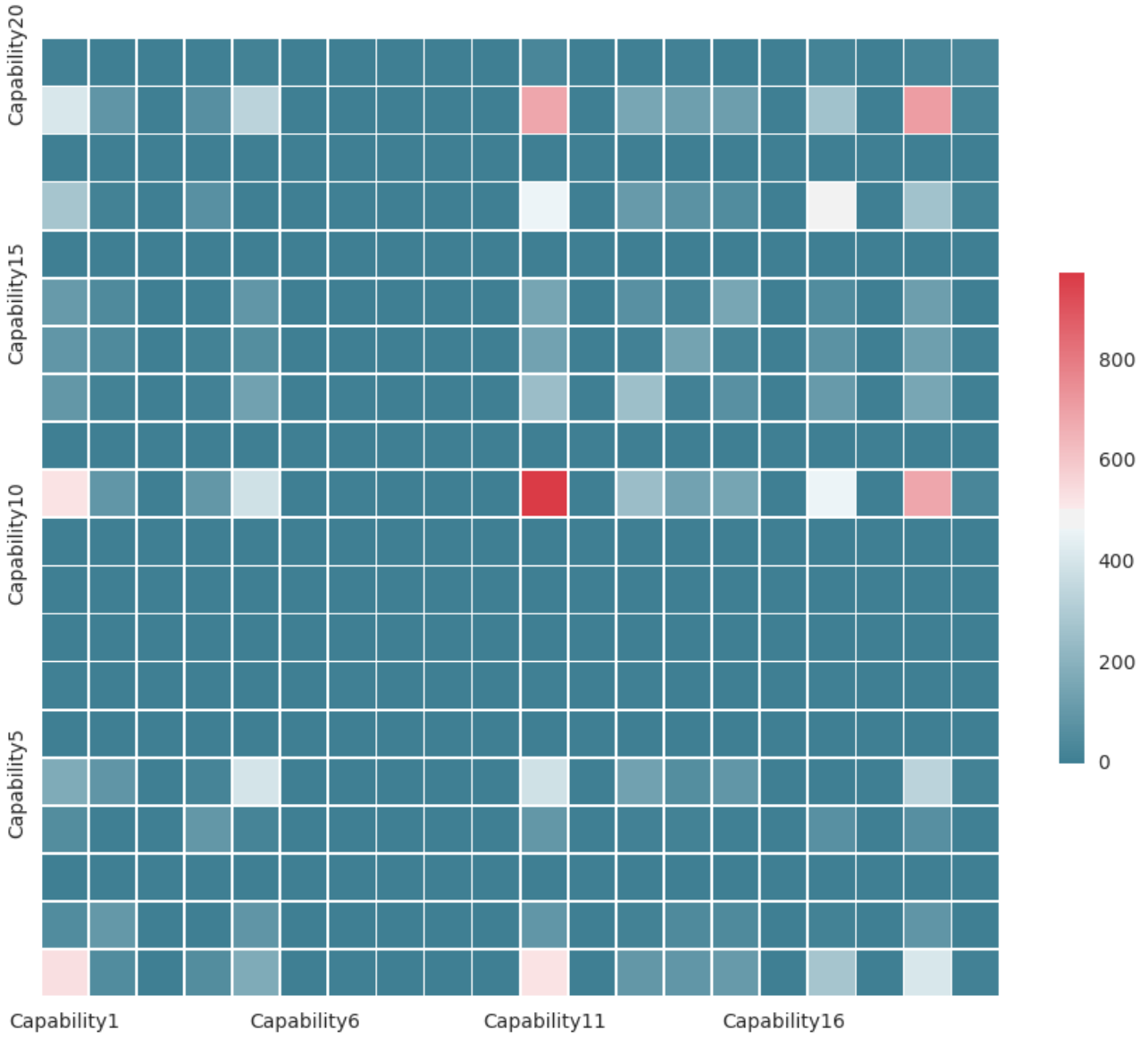

Figure 13. Capability Co-Occurrence

If a specific function or capability is in this category and is determined to be of high value and thus represents a gap it suggests a need to put forth the effort to close the gap.

It is also possible to mix coordinates in a co-occurrence matrix visualization. The chart below shows capabilities vs. components

```
m.show_ga_capability_component_cooccurrence_matrix()
```

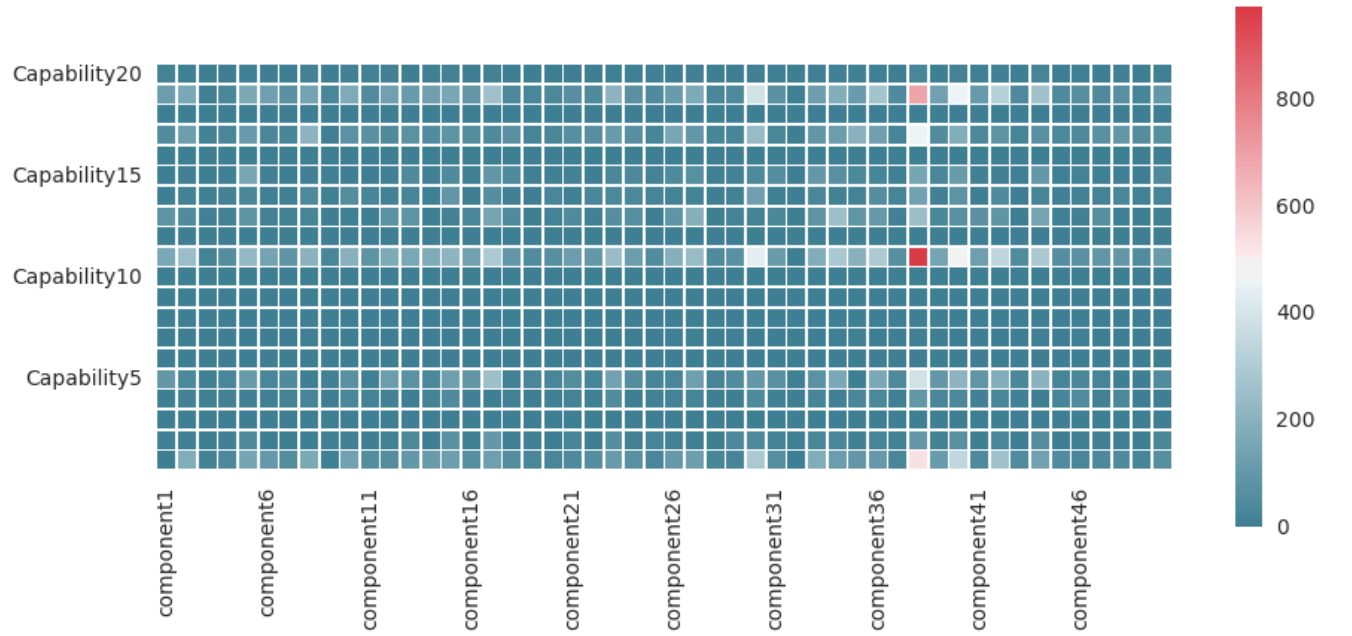

Figure 14. Capability vs. Component Co-Occurrence

In this view, many of the features of the component vs component and the capability vs capability matrixes are combined. Including the potential for revealing capability gaps coupled with traceability between capabilities and components. For example, this view could be useful for predicting useful components in order to achieve a target capability or this view could be used for identifying the overall usefulness of each of the components based on their overall co-occurrence across capabilities.

FUTURE PLANS

Additional work is required to formalize the methods described above related to time-based portfolio management. The strategies described to identify a preferred end state are well understood and formalized in the Jupyter Notebooks described in this report, but additional research is required to allow analyst to describe limitations on transitional states and have a framework provide candidate transformation paths for decision makers to review.

REFERENCES

[1] Browne, Daniel C., Shayler, Todd B., "Development and Application of FACT Portfolio Management Capability," Technical Report SERC-2015-TR-047-1. January 20, 2015.

[2] Jupyter Team, "Jupyter Documentation," http://jupyter.readthedocs.io/en/latest/install.html, Accessed September 12, 2016.

[3] Thomas, James J., and Kristin A. Cook. "A visual analytics agenda. *"Computer Graphics and Applications, IEEE* 26.1 (2006): 10-13.