META II

Complexity and Adaptability

Douglas Stuart and Raju Mattikalli
The Boeing Company

Daniel DeLaurentis
Purdue University

Jami Shah
Arizona State University

SEPTEMBER 2011
Final Report

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

1.0 SUMMARY

There were three important questions that we sought to answer in the DARPA Complexity and Adaptability metrics program. These were:

1. What are the complexity and adaptability metrics?
2. Do the metrics correlate with schedule, cost, or reliability?
3. And, can they be used to predict schedule given a conceptual design?
4. Are they useful as a relative metric comparing alternate designs?

In this summary we address each of these questions.

There is a large body of research on complexity. Much of this work has focused on complexity in mechanical design and software and is of keen academic interest. Everyone has a notion of what complexity and adaptability mean, and has the perception that intuitive views of complexity translate into longer schedules, greater costs, and less reliable systems. Our objective here was to not debate formal definitions, but instead to derive a set of observable metrics that are computable from a design and determine if the metrics, or combinations of them, can be correlated with observed data including schedule, cost, and reliability. This is the fundamental question, and our goal then is to find observable parameters that be computed from a design that have good correlation with observed cost, schedule, or reliability. Our approach to answer these questions was simple.

- Develop the metrics
- Select candidate programs that have design, cost, schedule, and reliability data
- Compute the metrics for each system
- Perform statistical analysis to identify correlations between computed metrics and the observed behavior of cost, schedule, and reliability
- Let the data speak for itself

1.1. Question 1

What are the complexity and adaptability metrics?

Answer: We have developed an extensive set of metrics (twenty or more) that are computable from a system design. One nice feature is that the metrics reflect an intuitive notion of system complexity. They include numbers of nodes, numbers of edges and loops in the system architecture, and strength of interactions between nodes, etc. The metrics are listed in Table 1.

Table 1 Selected Complexity and Adaptability Measures

Measures	Participates in Peak Labor	Participates in Schedule	Participates in Complexity	Participates in Adaptability
Coupling	🟩	🟩		🟩
Cycles				
Nodes	🟩		🟩	🟩
Edges	🟩		🟩	🟩
Average Node Weight				
Average Link Weight				
Average Weighted Node Degree				
Maximum Weighted Node Degree				
Weighted Connectedness				
Weighted Information Content				
Normalized Weighted Information Content				
Average Weighted Clustering Coefficient				
Number of Nodes and Links				
Weighted Mobility				
Links		🟩		🟩
Average Node Degree		🟩		
Maximum Node Degree	🟩			🟩
Connectedness		🟩	🟩	
Information Content	🟩	🟩		
Normalized Information Content		🟩	🟩	
Average Clustering Coefficient	🟩	🟩	🟩	🟩
Node Weight + Link Weight				
Mobility	🟩			🟩
Total Node Weight				
Total Link Weight				
Modularity		🟩		🟩
Integration Complexity				
System Modularity				

1.2. Questions 2 and 3.

Do the metrics correlate with schedule, cost, or reliability? And, can they be used to predict schedule given a conceptual design?

Answer: The short answer is that the data shows that there are strong correlations between combinations of individual metrics and peak labor and schedule for the system. The results are shown in Table 2which shows the functional relationships computed and the strength of correlation.

Table 2 Functional Relationships Between Measures and Peak Labor and Schedule

Terms in Model				
		BCA Systems	RMSE	Adjusted R-Squared
Peak Labor	1 - Term	13.2285 + 0.0306 * Nodes	19.75	0.32
	2-Terms	18.4768 - 71.303 * AvgClusteringCoefficient * AvgClusteringCoefficient + 0.20381* Nodes * AvgClusteringCoefficient	15.63	0.57
	3-Terms	20.1298 -140.873 * AvgClusteringCoefficient * AvgClusteringCoefficient + 0.47103 * Nodes * AvgClusteringCoefficient - 0.02673 * Nodes * MaxNodeDegree	12.44	0.73
Schedule	1 - Term	-1.1689* LN(NormalizedInfoContent)	1.26	0.94
	2-Terms	660.29 * NormalizedInfoContent * AvgClusteringCoefficient - 0.2332 * LN(Links) * LN(AvgClusteringCoefficient)	1.12	0.95
	3-Terms	879.88* NormalizedInfoContent * AvgClusteringCoefficient - 1.9065 * LN(AvgNodeDegree) * LN(AvgClusteringCoefficient) - 17.78 * AvgNodeDegree * Connectedness	0.66	0.98
		BDS Systems	RMSE	Adjusted R-Squared
Peak Labor	1 - Term	-617.6944 + 145.69 * LN(Edge)	55.95	0.76
	2-Terms	-773.25 + 234.0793 * LN(Edge) - 91.1 * LN(MaxNodeDegree)	47.13	0.83
	3-Terms	-425.3753 - 12.0086 * LN(Coupling) * LN(Nodes) + 42.8054 * LN(Edges) * LN(IvdDisp) - 34.6139 * LN(MaxNodeDegree) * LN(Mobility)	36.13	0.90
Schedule	1 - Term	0.3082* LN(Coupling) * LN(AvgNodeDegree)	0.67	0.96
	2-Terms	0.3753* LN(Coupling) * LN(AvgNodeDegree) - 5.677 * AvgClusteringCoefficient * Modularity	0.59	0.97
	3-Terms (No Intercept)	0.3318* LN(Coupling) * LN(AvgNodeDegree) - 11.67 * AvgClusteringCoefficient * Modularity + 0.5395 * AvgNodeDegree * Modularity	0.55	0.97
	3-Terms	0.6887 + 21.486* AvgClusteringCoefficient - 5.56 * AvgNodeDegree * AvgClusteringCoefficient + 0.11091 * IvdDisp * NormalizedInfoContent	0.37	0.93

The results were based on a set of candidate programs from Boeing Commercial Aircraft (BCA) and Boeing Defense Space, and Security (BDS). The BCA data included subsystems for the 787 Dreamliner. The advantage of the BCA data was that design information was directly derivable from the Product Data Manager (PDM) system. This enabled specific metrics to be computed without human intervention and furnished a set of data that was internally consistent. Boeing used a number of BDS systems and subsystems as sources of data. Whereas with BCA systems, much of the data was extractable from a PDM, significant manual effort including interviews with chief engineers was required to capture the design information from which the metrics were computed. This was performed by a single engineer using the same methodology to provide consistency in data.

We also discovered that there was only weak correlation with individual measures for both peak labor and schedule.

There are other important points. While we have developed functional relationships between peak labor and schedule and the metrics for commercial and military systems, it is not a one size (or equation) fits all. The combinations of metrics that have highest correlation include some common elements (e.g. MaxNodeDegree) but also have important differences. This may be an artifact of the data used in the calibration, but it also provides a caution on too much generalization. Also, when we compute complexity metrics and relate them to peak labor or schedule, we also need to remember that the relationship may not extend across system types. The BCA systems were primarily avionics systems. The BDS systems included aircraft, unmanned systems, and avionics systems. We do not believe that comparing complexity or adaptability metrics between system types is appropriate.

There are several observations to make.

For each program we have collected the following information: 1) Peak Labor; and 2) Schedule. Reliability turned out to be a poor observable for the set of systems available for calibration, since most if not all were flight/safety/mission critical systems, and accordingly the set of systems had almost uniformly very high reliability, and so did not provide enough variation for analysis. We have used peak labor (number of individuals working on the activity) as a proxy for cost. Pure cost data has a number of disadvantages. Costs are not necessarily directly comparable across different programs without significant adjustment (i.e., inflation, location …). Also, cost is not available except at a very summary level and as a result doesn't provide a good differentiator for subsystems. Peak labor on the subsystem is an observable and is highly correlated with cost. Peak labor is an indicator of the level of difficulty in product development and is a good representative of project size.

Schedule for major subsystems has turned out to be a more challenging as an observable than expected. Although schedule data is collected, too often the schedule for subsystems is not simply based on the amount of time required to complete it. It is often driven by other considerations such as: 1) Date specified by Government; 2) Availability of systems that it will be integrated with; or other factors. Small Diameter Bomb (SDB) had a two year development schedule, but it was specified by the terms of the contract and not driven by how quickly the work could be completed. The schedule also reflected timelines for integration of the weapon with other aircraft – driven more by aircraft and range availability than by readiness of SDB.

1.3. Question 4

Are they useful as a relative metric comparing alternate designs?

Answer: The metrics do appear to provide a basis for comparing alternate designs. We have identified functional relationships between the metrics and schedule and peak labor. The measures in Table 1can all be calculated for candidate designs, so that the functions representing peak labor and schedule can also be calculated, so that the results can be used to compare the design alternatives.

In addition to calibration against historical project data, we also explored the utility of the metrics in comparing design alternatives by conducting a variety of experiments that attempted to do just that. The selected measures were implemented in a prototype complexity and adaptability metric tool, which was integrated with the Cyber-Physical Systems Modeling Language (CyPhyML) modeling language, one of the META design languages and tool suites being developed concurrently by the Vanderbilt META Design Language and Design Flow teams. An infantry fighting vehicle (IFV) challenge problem was developed that included a design space represented in CyPhyML. The DESERT tool from the Vanderbilt tool chain was then used to generate 522 feasible candidate designs from the design space. The prototype metrics tool was then used to calculate the various Table 3 measures for all of the candidates, demonstrating the viability of using the metrics in the META design process.

Also, the relationships we identified through calibration provide a basis for assessing the impact in changes in the various metrics on the various observables. One can look at the various partial derivatives of the calibration functions to predict the change in the observables due to changes in the individual measures. Though there are limitations to this approach due to the uncertainties in the calibration function, and the differences between the calibrated functions across the two data sets, the presence of several of the individual measures in multiple calibration functions (as seen Table 1) argue for the reliability of those metrics as predictors, if not of the precise numerical formulation.

We have therefore shown that, within the limits discussed above, all of the questions we have posed above can be answered in the affirmative.

2.0 INTRODUCTION

2.1. Program Overview

The work reported herein was performed by the Boeing Company under Air Force Research Laboratory (AFRL) Contract FA8650-10-C-7085, "META II Metrics of Complexity and Adaptability." The Period of Performance (PoP) of the effort was 28 September 2010 to 30 September 2011. The AFRL Contract Technical Representative was Andrew Fleming of the AFRL Propulsion Directorate, Power Division, Energy Optimization and Assessment Branch (AFRL/RZPA). The Boeing Program Manager was Patrick J. Stokes, Dr. Douglas Stuart was Principal Investigator (PI). Subcontractors were Arizona State University, with PI Dr. Jami Shah, and Purdue University with PI Dr. Daniel DeLaurentis.

2.2. Goals

There is a vast literature on complexity. Research has produced a host of metrics for measuring complexity of systems. Adaptability has been less studied, but there are still a wide range of candidate adaptability metrics. In the course of our complexity and adaptability metrics program we have examined a wide spectrum of the existing metrics, and developed some novel metrics, and conducted experiments to determine those metrics that are most promising, in the sense of most likely to yield actionable predictions of their impact on system development. It is our hypothesis that the metrics will provide a computable metric that can inform stakeholder design tradeoff decisions, along with more traditional performance metrics. Such metrics will support quantitative design decisions that can trade different design choices based on quantitative metrics. One example would be quantitatively comparing the complexity driven impact on schedule of incremental performance differences between potential system designs. In order to support the sue of complexity and adaptability metrics in such a role, our work will has sought to answer three questions: 1) What are the metrics; 2) Can they be calculated for alternate designs; 3) Are they a basis for accurate prediction of programmatic data, specifically schedule, or are they akin to past performance disclaimers for a mutual fund – "not necessarily indicative of future results".

2.3. Team

The Boeing team consists of Boeing Arizona St University, and Purdue University. Dr. Douglas Stuart, Boeing, is the Principal Investigator. Dr. Raju Mattikalli, also of Boeing, has led both Metric Calibration and Experimentation The Boeing Company is a leading aerospace corporation with both commercial and military programs. Air vehicle programs range from small unmanned vehicles such as ScanEagle and Integrator to large fighter and transport aircraft. Boeing is also involved in numerous ground vehicle programs, ranging from Avenger through Brigade Combat Team Modernization. Boeing has a wealth of data covering multiple programs including cost, schedule, and design data that was used in calibration of metrics.

Boeing team members also included Dr. David Corman, Dr. Sabyasachi Basu, Dr. Rainer Romatka, Dr. Gregory Robel, Ron Howard, Tom Herm, Patrick Goertzen, James Meany, Lou Pape, Robert Scheurer, Donald Wilkins, Patrick Cassidy, Scott Boskovich, Jeff Holland, Mark Williams, Dr. Arnold Nordsieck, and Tom Barnett.

Dr. Jami Shah of ASU's Design Automation Lab (DAL) led the ASU element of the team. Dr. Shah and the DAL have been at the forefront of design and manufacturing research for the past 25 years. Professor Shah is the Director of this lab. The charter of the lab is to work on technologies that result in closer interaction between design and manufacturing. The lab is conducting active research programs in Intelligent CAD systems, Concurrent Engineering, Design for Manufacturing, and Tolerance Modeling. Previous studies related to META metrics development have included mechanical design complexity metrics for size, coupling and solvability, parametric design rating systems, domain independent manufacturability evaluation shell, and design exemplar networks to encode standard design features and procedures.

ASU team members also included Srinath Balaji, Prahsant Mohan, Gurpreet Singh, Zihan Zhang, Mahmoud Dinar and Xiang Ke.

Dr Daniel DeLaurentis of the Purdue Aerospace Systems group within the School of Aeronautics and Astronautics led the Purdue element of the team. Dr. DeLaurentis has been conducting cutting edge research on the design and operation of a wide variety of aerospace systems (and system-of-systems). In particular, he and his team have been leading the efforts to confront issues of complexity and uncertainty in conception, design, and development of networked Cyber-Physical Systems (CPS). Leveraging his background in probabilistic robust design and multidisciplinary analysis methods for aircraft design, in particular, his work studying network theoretic analysis to understand the topology of interactions within a system and between a system and other systems in its environment, Dr. DeLaurentis' group has been developing rigorous mathematical approaches for characterizing the behavior of complex systems and an approach for effectively using complexity in design trade-offs.

Purdue team members also included Shashank Tamaskar and Kartavya Neema.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1. Approach

Our overall approach is summarized in Figure 1. An initial set of complexity and adaptability metrics spanning the Adaptive Vehicle Make (AVM) domain was identified and/or developed. We then conducted a variety of experiments with these metrics to identify those that seemed likely to be relevant to the AVM domain. We then began the process of calibrating those selected metrics using various data sets, including design and schedule data from existing Boeing programs. We also began the development of a prototype tool for calculating the calibrated metrics from design artifacts. In the course of carrying out our approach, we collaborated with other research teams to understand the various design languages and flows and their impact on the metrics we have developed and selected, and to understand the relationships of the various proposed metrics. This has included integrating our metrics with the Vanderbilt Design Flow and Modeling Language teams' CyPhyML modeling language.

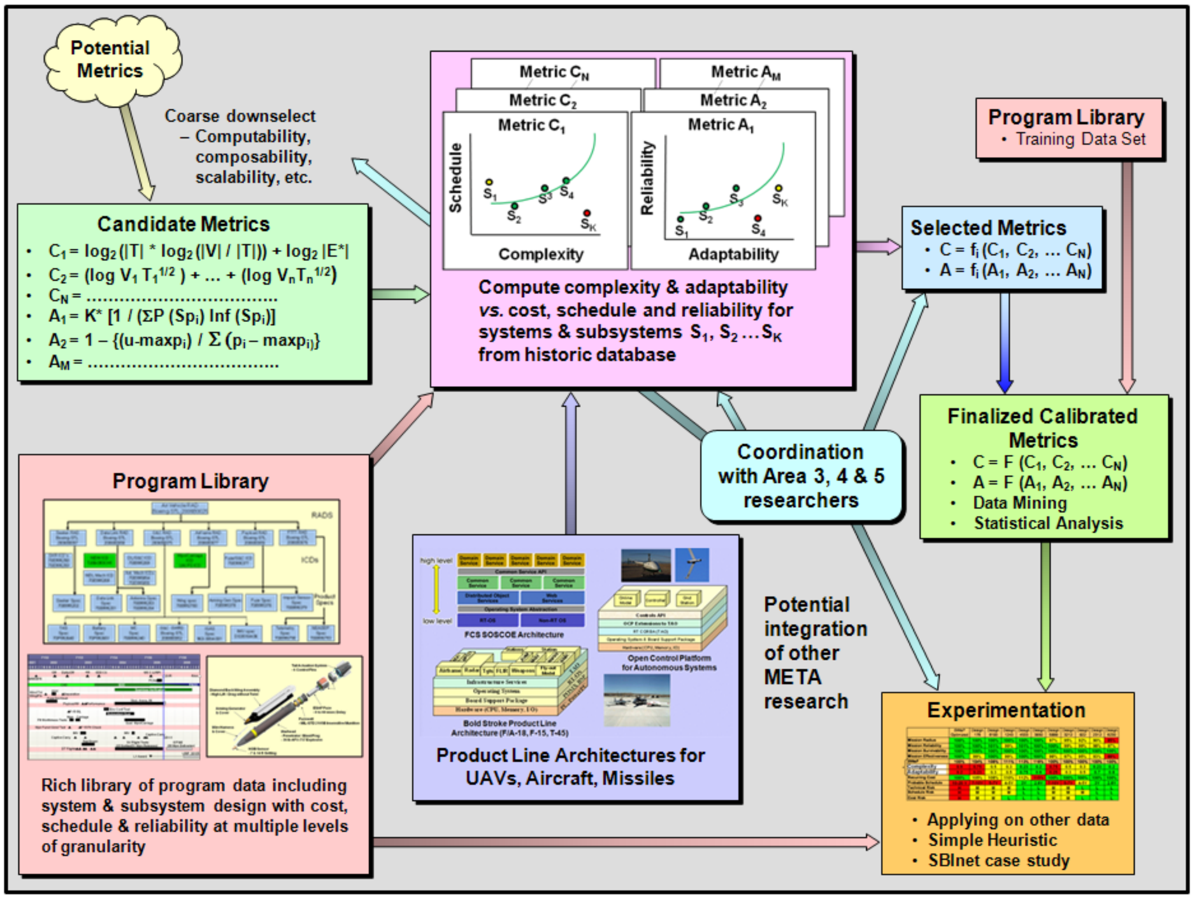

Potential Metrics
Coarse downselect
– Computability, composability, scalability, etc.
Candidate Metrics
$C_1 = \log_2(|T| * \log_2(|V| / |T|)) + \log_2 |E^*|$
$C_2 = (\log V_1 T_1^{1/2}) + \ldots + (\log V_n T_n^{1/2})$
$C_N = \ldots$
$A_1 = K^* [1 / (\Sigma P (Sp_i) \text{ Inf } (Sp_i)]$
$A_2 = 1 - \{(u\text{-max}p_i) / \Sigma (p_i - \text{max}p_{ij})\}$
$A_M = \ldots$
Metric C_N
Metric C_2
Metric C_1
Schedule
Complexity
Metric A_M
Metric A_2
Metric A_1
Reliability
Adaptability
S_1 S_2 S_3 S_4 S_K
Compute complexity & adaptability *vs.* cost, schedule and reliability for systems & subsystems $S_1, S_2 \ldots S_K$ from historic database
Program Library
• Training Data Set
Selected Metrics
• $C = f_i (C_1, C_2, \ldots C_N)$
• $A = f_i (A_1, A_2, \ldots A_N)$
Finalized Calibrated Metrics
• $C = F (C_1, C_2, \ldots C_N)$
• $A = F (A_1, A_2, \ldots A_N)$
• Data Mining
• Statistical Analysis
Coordination with Area 3, 4 & 5 researchers
Program Library
Rich library of program data including system & subsystem design with cost, schedule & reliability at multiple levels of granularity
FCS SOSCOE Architecture
Open Control Platform for Autonomous Systems
Bold Stroke Product Line Architecture (F/A-18, F-15, T-45)
Product Line Architectures for UAVs, Aircraft, Missiles
Potential integration of other META research
Experimentation
• Applying on other data
• Simple Heuristic
• SBInet case study

Figure 1 Approach to Developing and Calibrating Metrics

4.0 RESULTS AND DISCUSSION

4.1. Initial Candidate Metrics

In this section we present the initial candidate complexity and adaptability metrics we have developed over the course of the program. These metrics come from a variety of sources, and we have performed various experiments with a subset of the candidate metrics to identify those which will be used to make up our final complexity and adaptability metrics.

4.1.1. Complexity

There are a variety of complexity metrics that have been identified as candidates for a META complexity metric, or as constituent measures within a META complexity metric. This section describes various candidate complexity metrics that we have identified, drawn from a diverse set of sources.

Entropy of a system is frequently used as a candidate complexity metric. Various formulations of entropy have been used in domains ranging from information theory to Suh's axiomatic design theory. One common approach is to look at the probability of satisfying functional requirements. The information content is then defined in terms of the probability of satisfying a particular functional requirement (Equation (1)). For satisfying a set of requirements, the formula is modified to include the joint probability of satisfying all the requirements. The figure addresses the case of uncoupled requirements. In the case of coupled requirements, the joint probability becomes a conditional probability.

$$\begin{aligned} I &= \log_e\left(\frac{1}{P}\right) = -\log_e P \\ I_{sys} &= -\log_e P \; where \; P_m = \prod_i P_i \\ I_{sys} &= \sum_i \log_e\left(\frac{1}{P_i}\right) = -\sum_i \log_e P_i \end{aligned} \tag{1}$$

Other entropy based measures have looked at the probability of producing a design by considering the size of the design space. These include the metrics proposed by Braha [1] and Malmon [2] (Equation (2)) and Summers and Shah [3] (Equation (3)). Suh [4] further refined his concept of information complexity by introducing the notions of real and imaginary complexity (Equation (4)). Real complexity is the "true" information content of the system, reflecting perfect knowledge on the part of the designer, while imaginary complexity is that part of the complexity due to lack of knowledge on the designer's part.

$$H = \sum_i \frac{1}{\lambda^\Delta} \ln\left|\frac{1}{\lambda^\Delta}\right| \qquad (2)$$

$$Cx_{siz\_prob} = \{(M^U + C^U) \times \ln|idv + ddv + dr + mg|\} \qquad (3)$$

$$C_R = \sum_i \ln\left|\frac{1}{P_i}\right| \quad C_i = \ln\frac{1}{P_{Prob}} = \ln(n!) \qquad (4)$$

Another source of complexity is coupling between elements of a system, design, or process. Interactions are a major source of complexity in and of themselves, and are also generally responsible for emergent behavior, which is usually not predicted or predictable. Measures for coupling therefore represent a potentially fertile ground for META complexity metrics. Coupling is also a factor in adaptability as will be discussed later. Note too that there are also network centric coupling metrics for complexity which will be discussed later in this section. One coupling metric is due to El-Haik [5] and Yang [6] and is shown in Equation (5), where *DP* are the design parameters, *h* is the information measure, and *ρ* is the correlation coefficient between two design parameters.

$$h(\phi(DP_i, DP_k)) = \ln(2\pi e\sqrt{1 - \rho^2}) \qquad (5)$$

Many design, development, and manufacturing artifacts and concepts can be represented as graphs. These range from system architectures where the nodes in the graph represent components and the edges in the graph represent interfaces between the components to manufacturing process graphs where the nodes represent manufacturing operations and the edges represent the flow of work. Graphs and networks have also formed a fertile basis for the study of complex adaptive systems, with the resultant development of characterizations of system complexity in terms of the graph representations. There are accordingly a variety of metrics from network and complex adaptive systems theory that could be used as the basis for META complexity metrics (and adaptability metrics) when applied to graph based system representations. Some are relatively straightforward. The number of nodes in the graph, and the number of edges provide a raw indication of system scale.

Other fairly simple metrics address the coarse connectivity of the graph. The degree of a node is the number of links incident to it. The average node degree then is an indication of the average connectedness. Other degree related metrics include the highest degree of all nodes (corresponding to hubs in the graph), and degree distributions. For weighted graphs, the weighted degree (strength) sums link weights associated with nodes.

More sophisticated connectivity metrics are also possible and could provide deeper insights into the structure (and complexity) of the underlying system. The clustering coefficient is the ratio of the number of triangles in the graph to the number of triples centered on a node (in Figure 2, the clustering coefficient for node ‘3’ is 1/3). It corresponds to the “local” cohesiveness of the network and can be viewed as indicative of robustness or fault tolerance.

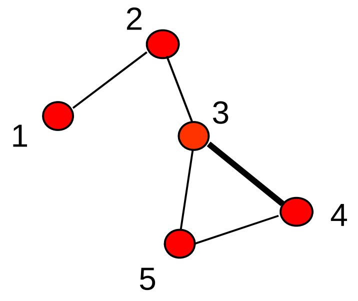


Figure 2 Clustering Coefficient

Centrality measures attempt to directly measure the importance of individual nodes by their impact on the connectedness of other nodes. For example, betweenness centrality measures the fraction of shortest paths through a node and can identify potential bottlenecks. There are variations of betweenness centrality such as eigenvector centrality which rates the importance of a node based on the importance of nodes to which it is connected, similar to the Google page rank algorithm.

Coupling complexity is another network based metric. It captures the product of the topology of interactions within the system, including accounting for directivity and weights within a network. It also tests for the presence of feedback loops within the system. Another key feature of this metric for coupling is its ability to measure coupling between nodes, which are not directly connected. Sometimes, design changes in one part of the system often affect other parts of the system not directly connected with it. Our proposed metric apart from accounting for direct connectivity between the components also accounts for all the indirect interactions, which are associated with a particular link. The algorithm for computing the coupling metric is captured in Figure 3.

In the algorithm, c and j denotes the number of cycle and size of the cycle respectively. W_i denotes the weights of the links of the cycle. m is the number of links which are not the part of any cycle. W_k denotes the weights of those links that are not the part of any cycle.

1. Represent the system using directed and weighted structural graph of the system.
2. To capture coupling between indirectly connected nodes, we propose to calculate how frequently each links of the network are used while finding all the paths to connects all the pairs of the nodes in the network. Any indirectly connected node of the network can only interact using the paths laid between them. This step will ensure higher weightage to links that are used larger number of times and thus capture coupling between indirectly connected nodes.
3. Define refined weights of the system by multiplying the weights of the network by the frequency of each link calculated in step 2.
4. Calculate coupling complexity using the formula.

$$\text{Coupling Complexity} = \sum_{n=1}^{c} j_n \left(\sum_{i=1}^{n} W_i \right)_n + \sum_{k=1}^{m} W_k$$

Figure 3 Coupling Complexity Algorithm

Other graph related metrics include connectedness (Equation (6)) and the normalized information content of a node (Equation (7)). At the graph level, the Mobility metric (3(N-L-1)+N for a graph with N nodes and L edges) attempts to capture the degree of freedom available to a node. For a final network based complexity metric, consider solvability, which is greatly aided by decomposability. Bridge links indicate the decomposability of a graph. A bridge link connects two different sub graphs of a structure. On removing a bridge link, the graph will split into two sub graphs. Alternatively, we can consider bridge nodes instead of links. However, not all bridge links or nodes are equally important; the ratio of the size of the resulting subgraphs, s_1 and s_2, indicate the importance. The best ratio Ns_1/Ns_2 is 1. Leaf nodes form trivial bridges where the ratio is (N -1)/N. The bridge impact factor is defined as the average subgraph ratio taken over all bridges, excluding trivial bridges.

$$C = \frac{2L}{N(N-1)} \tag{6}$$

$$NI_{VD} = \frac{I_{VD}}{(I_{VD})_{max}} \quad where\,(I_{VD})_{max} = N(N-1)\log(N-1) \tag{7}$$

4.1.2. Adaptability

Common definitions of adaptability include to make fit for a new purpose or situation often by modification, and modification according to circumstances; adjustment to environment. In the context of product design, we can cast the above definitions into four categories, reliability, robustness, modularity, and requirements adaptability.

Adaptability to changes internal to a product; this may include the probability of recovering from or preventing malfunctions, the ability to recover from partial failure, or reliability with respect to potential failure modes. This type of adaptability is often referred to as reliability.

Adaptability to uncontrollable or unexpected external variations, such as changes in the operating environment, manufacturing, supply chain, availability of materials. This type of adaptability is often measured by a robustness metric.

Adaptability to product evolution or variety. This may either be in the form of mix-and-match modules to produce greater product variety, or a platform based design with optional add-ons for additional functions, technology upgrade or future capacity scaling. This type of adaptability is often referred to as modularity.

Adaptability to new requirements (e.g., variations in mission parameters, payload, range, weapons systems). There are no conventional measures for this type of adaptability, so we will refer to it as requirements adaptability.

Since methods for Type 1 and 2 above are well established in industry we will not dwell on them very much in this report. For the sake of completeness we will give a quick overview of Reliability and Robustness, but will focus on modularity and adaptability to requirements change. Note, too, that several complexity metrics related to modularity also apply to adaptability (for example the coupling and modularity metrics above).

4.1.2.1. Reliability

Reliability is the probability that an element, device or system will not fail to perform its intended function within specified limits for a given period of time in a specified environment, so that $R = 1 - P$, where P is the probability of failure.

Determination of reliability is a routine part of all product design. Particularly in air and space vehicle design very high reliabilities are required. This is achieved by designing fail-safe, redundant and parallel subsystems, Figure 4, which adds complexity to enhance reliability.

System reliability is computed from component reliability and system architecture, modeled as a combination of parallel and series subsystems. A series combination is defined by the criterion: “failure of one element causes system failure”. Therefore, a series system reliability is Rn = Π_i R(i)

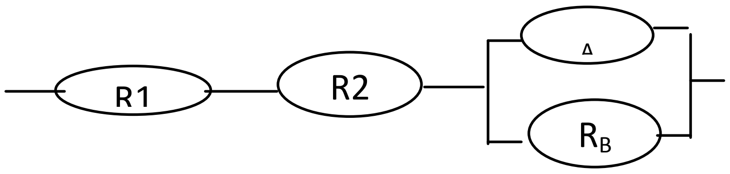


Figure 4 Reliability Through Redundancy

A Parallel combination is defined as “all elements must fail to cause system failure”. For a parallel system consisting of A and B is R(AB)= 1 - P(A)P(B).

Failure analysis of components involves identification of potential failure modes, calculation of failure conditions, and use of material properties, analytical or empirical data to predict the probability of each failure mode.

FMEA is a procedure for analysis of potential failure modes within a system for classification by severity or determination of the effect of failures on the system. Developed originally as MIL-P-1629 by the military. Used by NASA in the Apollo project. Uses the metric called Risk Priority Number (RPN), Figure 5defined as the product of the probability of Occurrence x Severity of failure (cost, degree of injury, property damage) x Detectability (to avoid failure).

Function	Failure Mode	Potential cause	Occurrence rating	Local effects	End effects	Severity rating	Detection method	Detectability rating	RPN

Figure 5 Risk Priority Number

A number of spreadsheets are in common use for computing RPN metric in association with predefined scales (1 -10) for measuring severity and detectability. The probability of failure occurrence is also scaled 1 to 10. For example, 1 in 1,500,000 probability is deemed remote and assigned a value of 1 while 1 in 2 is deemed "very high" and assigned 10. Thus, RPN can take values from 1 to 1000, the higher the number the greater the risk. An appropriate threshold value based on context has to be chosen to define acceptable risk.

4.1.2.2. Robustness

Robustness is defined as the degree to which a product's "quality" is insensitive to variations in manufacture and operating conditions. Quality can be any set of product attributes that are considered important to the customer. Robust design is a philosophy that recognizes that there are variables a designer can control (**S**ignal) and ones he cannot (**N**oise). Unlike traditional design where the objective function is defined in terms of only the Signal variables, Robust Design uses S/N ratios. The difference between the two approaches is illustrated in Figure 6. While conventional design looks for the highest peaks, Robust Design looks for the highest mesas – regions where the performance does not fall off due to small variations in design variables (due to noise).

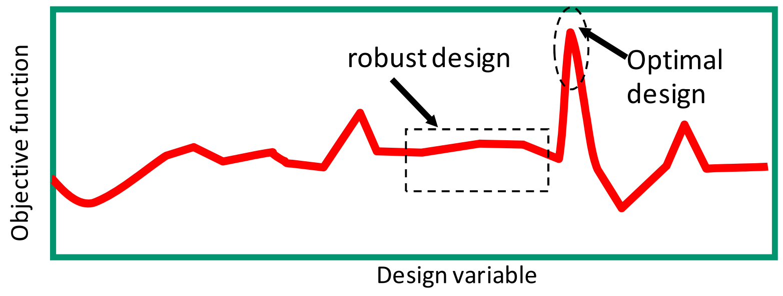


Figure 6 Robust Design

This has important implications in META. Instead of choosing a design alternative that is highest performing we might choose a lesser design in order to make it more adaptable to operating or manufacturing variables that we do not control. It is hard for engineers to walk away from a better design unless we redefine the measure of goodness. We also note that this metric does not make explicit what other costs you are paying for robustness.

Although, S/N ratios can be used as objective functions in any optimization method, such as gradient based methods or GA/GP, Taguchi's [7] popular implementation uses Design of Experiments (DOE) approach instead. It involves two phases: (Nominal) Parameter Design and Tolerance Design. The first involves factorial DOE to sample the design space and find promising regions. The second involves analysis of small variations (tolerances) from nominal values. Figure 7 shows a conceptual example of sampling parameter design space (the red dots are used sampling points; grey are unused in a factorial experiment). The inset square represents detailed investigation around a nominal design point. Taguchi proposed several empirical S/N functions as metrics, such as the one in Equation (8), depending on whether the design objective was to maximize, minimize or achieve a target value. These functions are controversial as they are without any theoretical basis.

$$\frac{S}{N} = 10\,log_{10}\left(\frac{y_m^2}{s^2}\right)\; s^2 = \text{ variance of } y,\; y_m = \text{target value} \qquad (8)$$

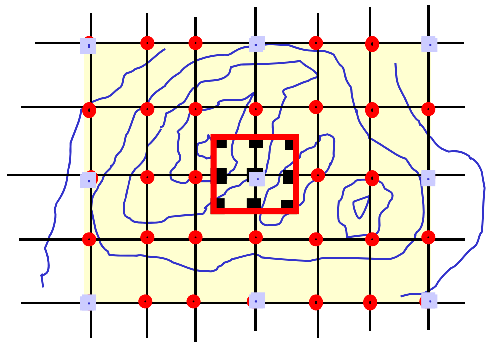
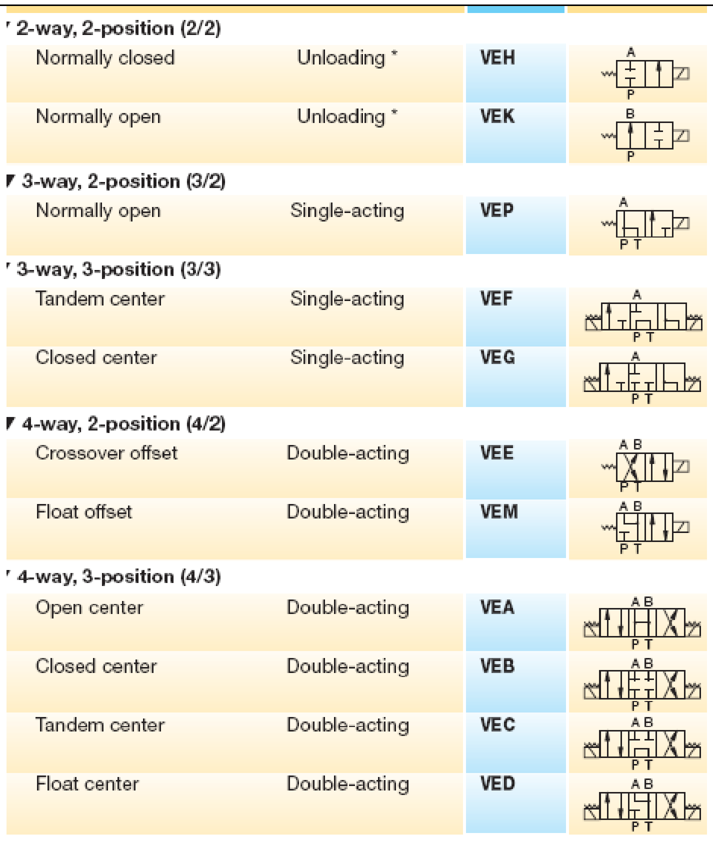


Figure 7 Example Taguchi Function

4.1.2.3. Modularity

To understand adaptability metrics geared towards measurement of modularity, we need to understand not only the technical architecture and design methods of modular products but also the economic motivation. Modular products are assemblies of functional units (modules). Modular design is motivated when product variants are planned or anticipated. In the example shown in Figure 8, a variety of Directional Control Valves (DCV) can be synthesized from a few basic valve modules. Modularity in product architectures offers a number of advantages. Commonality amongst product families can be exploited by reducing the variety of parts that need to be designed and manufactured, which in turn may lead to lower inventory. New products or variants could be designed in shorter time by re-using existing modules. Technologically outdated modules can be easily replaced provided the new module meets form, fit and functional requirements imposed by the legacy system. This can be done by carefully designing standardized interfaces between modules. The drawback for modular systems is that the modules cannot be optimized for a specific system as they need to serve a larger spectrum of products. Other drawbacks include greater design effort and higher initial investment. Empirical studies of mechanical products have shown that most modular systems are typically heavier and have larger volume than monolithic systems.



2-way, 2-position (2/2)		
Normally closed	Unloading *	**VEH**
Normally open	Unloading *	**VEK**
3-way, 2-position (3/2)		
Normally open	Single-acting	**VEP**
3-way, 3-position (3/3)		
Tandem center	Single-acting	**VEF**
Closed center	Single-acting	**VEG**
4-way, 2-position (4/2)		
Crossover offset	Double-acting	**VEE**
Float offset	Double-acting	**VEM**
4-way, 3-position (4/3)		
Open center	Double-acting	**VEA**
Closed center	Double-acting	**VEB**
Tandem center	Double-acting	**VEC**
Float center	Double-acting	**VED**

Figure 8 Hydraulic Directional Control Valves (from ENERPAC product catalog)

Functional modularity manifests itself as physical modularity. There are different types of functions in modular products: Basic, Interfacing, Expanded and Custom. Basic functions are essential to the system and are typically recurring. In the hydraulic valve example the basic functions are allowing the flow in one direction or the other, or stopping the flow. Interfacing functions are needed to combine modules together (locate, fasten, adapt mating elements). Optional modules may be needed for expanded functions. For example, we could combine DCV modules with FCV modules to control not only the direction but the flow rate. Custom functions are non-standard; may be implemented for particular customers and will usually not be modular because of limited re-use or need by other customers. In addition, one might leave a system open to additional unknown functions that may be needed in the future. Mixing and matching basic and optional modules yields functional variants. These are often also combined with size variants using the same type of functional module but sized differently. Therefore, modular combinatorial plans can give many size and functional variants. Moving beyond the initial planned variants may only require partial redesign since if some existing modules can be included in the new design. Such a product may be said to be adaptable to new requirements.

Metrics for modularity are typically based on functional coupling. We can take two different viewpoints. In one we look at the potential for modularity in a product and in the other at the extent of modularity that is actually implemented. A product can consist of 1: n modules; a module can consist of 1: m parts. One extreme is that the whole product is one module; the other extreme is that there are as many modules there are parts. This can provide a scale for measuring modularity as done in some metrics.

In modular design methods, one looks at the strength of coupling between design entities. The ones that have greater interactions amongst themselves than with those outside, are grouped together to form a potential module. We can look at entity relations at design variable level alone, function level alone or both. If we represent the functional coupling by a graph various network measures can be used. Components of a cluster are suitable for a module, because adaptations to the implied components will not cause numerous change impacts to further external components.

Looking only at product architecture to evaluate the effectiveness of modularity does not tell the whole story. The level of coverage of product variety is based on market demand, either current or projected by forecasts. The higher initial investment in design effort and cost can only be justified by conducting a study of return on investment.

4.1.2.3.1. Metrics based on DSM connectivity

In theory, modularity can be assessed by examining interactions between functional requirements (FR), coupling between design variables (DV), or connections between physical parts (PP). During conceptual design or redesign, one might be interested in looking for potential opportunities for clustering parts into modules. When looking at existing products one can assess the extent to which a product is modularized. One way to represent coupling is the Design Structure Matrix (DSM) which can be used at a single level (FR-FR, DV-DV, PP-PP) or across different levels (FR-DV, DV-PP, FR-PP), also called multi-domain matrices (MDM). In the former case, the matrices are square.

Most metrics found in the literature are based on DSM at the physical (PP) level only. A straightforward example is the Whitney Index [8] which is based on the ratio of the number of interactions #i (non-zero entries) in DSM to the size #e of the DSM. The purpose of the index is to reduce system complexity by modularizing. The interactions both within and across modules are taken into consideration in order to compute this metric, *WI = #i / #e.*

From empirical studies at General Motors, they found the ideal ratio to be 6.3. Deviation on either side is deemed less than ideal. This is a dimensionless ratio and simple to calculate. It is not very clear what is being measured. It also lacks any normalization for comparison with other measures. DSMs are in raw form, considering parts in isolation (non-modularized).

Ulrich [9] based his metric on the following observations about modular architectures: Chunks (of parts) implement one or a few functional elements in their entirety. The interactions between chunks are well defined and are generally fundamental to the primary functions of the product. Thus a modular architecture is one in which each functional element of the product is implemented by exactly one physical "chunk" and in which there are a few well-defined interactions between the chunks. Ulrich Modularity Index was thus defined as the ratio of the number of components #P to the number of functions #FR, *UI = #P / #FR.*

The ideal value is 1 (functional independence axiom?). Like WI this is dimensionless and simple to compute, although one can argue about the non-uniqueness of component and function decomposition needed. They provided empirical evidence of modularity from small electro-mechanical systems. Even though an ideal value 1 means the product is fully modular, it also means that as the number of functions increase, the number of parts also increase. Hence inventory costs are not considered. This metric can be calculated only late in the development cycle. Neither UI nor WI take into account manufacturing and assembly considerations or the nature of interfaces or the number of connections between parts.

4.1.2.3.2. Modularity metrics based on SVD of DSM

A singular value decomposition (SVD) of the DSM matrix reveals the singular values and the corresponding orthogonal Eigen vectors. Integral systems have one or more singular values that are very large and other values that are pretty small (Large variance). A completely modular system has very low singular value and shows a gradual decay of its singular values. The singular value modularity index (SMI), Equation (9) , proposed by Hölttä et al [10] measures the average, weighted decay rate of sorted singular values in the DSM. In the equation, N is the number of components, and the σ_i are the singular values of the DSM in decreasing order.

$$SMI\left(\sum DSM\right) = 1 - \frac{1}{N * \sigma_1} \sum_{i=1}^{N-1} \sigma_i(\sigma_i - \sigma_{i+1}) \qquad (9)$$

Since this metric is based on DSM and standard matrix methods, it is fast to compute even for large DSMs. The study provided empirical evidence based on simple electro-mechanical systems. Like UI this metric has an ideal value of 1 for a completely modular system. A fully modular design is not always desirable. Instead an optimal value needs to be defined. As an aside, since changes in requirements affect the interactions in the DSM matrix SMI could potentially compute "sensitivity". If the change in SMI is small with change in requirement then the DSM is not sensitive or if the change in SMI is high then the DSM is sensitive to requirements change.

Gershensen proposed a similar metric but with the difference that they account for modular arrangement already in place [11]. So they do not count interactions within module but only across modules. The measure calculates modularity by subtracting the averaged interactions external to modules from the averaged internal within modules. A design is said to be completely modular if all the interactions in a system exists within the modules. The modularity score for such an ideal system is 1. The common measure was created by testing and correcting the "best" representative measure selected from seven mainstream measures. The measure was validated based on contriving obviously modular product matrices that are scrambled from their known optimal modular states and then integrating this common measure with a modular product design method to redesign these scrambled matrices. It was verified that this common measure performs well in redesigning the scrambled matrices back to their known optimal modular states without rewarding redesigns that trend towards one large module or being affected by the different subjective scales used to measure component–component relationships.

Gerhensen's Modularity Index (GMI) is defined in Equation (10), where n_k and m_k are the index of the first and last components in the kth module, M is the total number of modules, N the total number of components, and R is the value of the ith row and jth column in the MDSM.

$$\frac{\sum_{k=1}^{M}\frac{\sum_{i=n_k}^{m_k}\sum_{j=n_k}^{m_k}R_{ij}}{(m_k-n_k+1)^2}-\sum_{k=1}^{M}\frac{\sum_{i=n_k}^{m_k}\left(\sum_{j=1}^{n_k-1}R_{ij}+\sum_{j=m_k+1}^{N}R_{ij}\right)}{(m_k-n_k+1)(N-m_k+n_k-1)}}{M} \qquad (10)$$

Gerhensen's formula is not mathematically SVD but conceptually measures the same thing because all the interactions in a DSM matrix are desired to be along the diagonal (within the modules). Interactions far from diagonal are penalized by both the SVD method and the Gerhensen's formula. Since DSMs must be organized by modules, this metric is sensitive to the choice of modular boundaries.

It can also be used in sensitivity analysis by changing the matrix inputs and then measuring the percent of output change over the original output. Sensitivity can be evaluated in three ways: the average value of change, the upper change limit, or the lower change limit. The average change gives a clearer picture of what is happening over the range of products.

4.1.2.3.3. Other design oriented modularity metrics

One of the weaknesses of all the metrics discussed so far is the lack of consideration of interfaces. Also, modularity is created in the context of product variety, which none of the above measures account for. Strong's metric [12] considers at least interfaces although not variety; it counts the variety of interfaces needed in a modular product. Their Interface Re-use Metric (IRM) is based on the ratio of interface types (#type) to total interfaces (#total): $\mathrm{IRM} = 1 - \frac{\#\,\text{type}}{\#\,\text{total}}$.

Interactions are expressed in the physical realm. Interactions between functions are not considered. The interface types are classified as shown in Table 3. They measure degree of modularity the same as UI but they add the IRM index.

Table 3 Physical Interactions

		Architecture Type	
		Base	Baseless
Interface Type	Unique Interface	Slot-Modular Cut-to-fit	Mixed-Modular
	Standard interface	Bus-Modular Component Swapping	Sectional Modular

Some researchers have represented product architectures graphically instead of using DSM and proposed using common network measures , such as Coupling Co-efficient, Betweenness (the number of times a vertex occurs on a geodesic, the short path connecting two vertices), Centrality (connectedness of each node [13]. In this approach, the distinctions between complexity and adaptability can become blurred.

Purdue has developed another modularity adaptability metric that blurs the line between complexity and adaptability results from considering the integration complexity of the system's subsystems. The coupling complexity metric defined previously considers the system as a whole. The new metric measures modularity by first considering the coupling complexity of the system as a whole (no decomposition of the system into subsystems) and then divides the system into two subsystems and calculates the complexity of the subsystems and their integration. Under this notion, the modularity of the system is equal to the summation of the subsystem's complexity and integration complexity.

Modularity is an important design characteristic that designers often seek to maximize in their realized systems. A modular design is one whose decomposition into subsystems is such that each subsystem acts independently with minimal influence from the other subsystems. Hence, the complexity of the system should decrease if we properly divide the system into subsystems and then integrate it. A coefficient of modularity (CM) is introduced to accommodate for this impact on complexity due to modularity. The CM represents the ratio of integration complexity and subsystem complexity and reflects the degree to which the decomposition of the system into subsystems is good or bad. Next, the "modified integration complexity" is defined as the product of integration complexity considering system as a whole (CI) and CM. Therefore, the new system modularity equals summation of subsystem complexity and modified integration complexity (CI_M) for a system with optimized modularity.

Decomposing a large system with thousands of components into optimal modules is a challenging endeavor. One means for the identification of modules is through use of Newman's algorithm [14], which is based on the notion of community structure within a network. Community structure is closely related to the notion of clustering; however it is not the same. A community is detected in a network based on the intuition that nodes within the same community are more densely connected as compared to that of inter community nodes. Newman's algorithm is a type of divisive algorithm that seeks to divide the network into communities by successively removing links. Links having the highest "betweenness centrality" (i.e., having a role in many shortest paths) are removed first followed by other links in descending order of the "betweenness." Once the links are removed and separate communities are identified, a term called modularity is calculated which represents how well the network is divided. The process of removing links, identifying communities and calculating modularity continues until modularity reaches a local maximum.

While this algorithm works well with social and biological networks that usually deal with simple associations, it does not do well with directed networks often found in engineering systems. Another shortcoming of this algorithm is that it ignores the presence of feedback loops in the system. As highlighted with the coupling complexity metric, feedback loops are significant drivers of complexity in any engineering system. Feedback loops or cycles are inherently indivisible and all the components within a cycle must be designed together. Hence, any optimal system decomposition should ensure that links associated with cycles are not divided; cycles should remain localized to individual modules.

An alternate approach was devised to overcome the shortcomings (for our applications) of Newman's method. This modified approach also relates naturally to the coupling and integration/modularity-based complexity metrics described in the previous sections. Two modifications distinguish the approach from that of Newman. The first is the approach to link removal, and the second defines the stopping criteria.

Links corresponding to highest betweenness centrality are removed if the following three conditions are satisfied. First, link removal does not cut links of the cycle. Second, link removal does not result in complexity of a subsystem to be less than the integration complexity. Third, link removal only occurs in the (currently) largest module. Our modified algorithm terminates when the integration complexity is greater than the average complexity of the subsystem.

The example in Figure 9 highlights the operation of the modified approach. Note that no cycles are disturbed in the identification of modules. Complexity of the network considering the system as a whole is 489 and the value obtained after optimal division into modules is 461.8. The

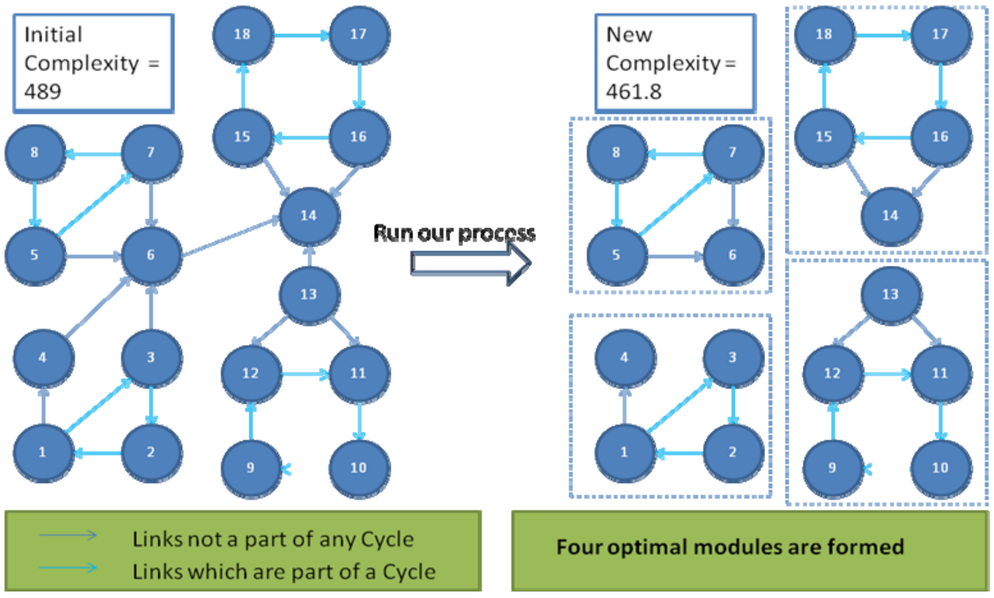

Initial Complexity = 489
Run our process
New Complexity = 461.8
Links not a part of any Cycle
Links which are part of a Cycle
Four optimal modules are formed

Figure 9 Example Illustrating the Modified Approach for Optimal Module Identification

relatively small change in complexity is just an artifact of this simple example that was already indicative of modularity initially.

4.1.2.3.4. Manufacturing Modularity metrics

Changes in product design can cause costly changes in manufacturing. The greater the extent of common parts in product variants, the lower the manufacturing cost due to larger volumes, fewer change-overs and less setup time. Based on this idea Martin & Ishii [15] did empirical studies of automobile instrument cluster variants to define three indices, Commonality, Differentiation and Setup. The Commonality Index (CI) (Equation (11)) is a measure of how well the design utilizes standardized parts. In the equation, u is the number of unique parts, and p_j is the number of parts in model j.

$$CI = I\frac{u - max\, p_j}{\sum_{j=1}^{v_n} p_j - max\, p_j} \tag{11}$$

The Differentiation Index (DI) (Equation (12)) measures the differentiation that occurs within the process flow. In the equation, v_i is the number of different products, n the number of processes, and v_n the final number of varieties. D_i is the average throughput time from process i to sale, and a_i is the value add of process i.

$$DI = \frac{\sum_{i=1}^{n} d_i v_i a_i}{n d_1 v_n \sum_{i=1}^{n} a_i} \tag{12}$$

The setup index SI (Equation (13)) is an indirect measure of how switchover costs contribute to overall product costs. In the equation, v_i is the number of products, c_j the cost of setup for process i, and c_j is the total cost of the jth product.

$$SI = \frac{\sum_{i=1}^{n} v_i c_i}{\sum_{j=1}^{v_n} c_j} \tag{13}$$

These metrics do not take into account standard materials, standard tools and fixtures. These aspects are included in Brill's metric [16] (Equation (14)). Essentially, products that can be manufactured by standard processes, on standard equipment without custom fixtures, tools, dies offer the greatest flexibility. $F_{M,S}$ is the flexibility of a machine relative to a task set, e is the efficiency of a machine for a task, and W is the importance of a task.

$$F_{M,S} = \frac{\int_{\tau \in S} e(M,\tau)\, dW(\tau)}{W(S)} \tag{14}$$

These metrics are manufacturing centric; design functions or assembly structure is not considered. Also, the procedure for determining weights is not clear. Much subjective assessment needs to be made.

4.1.2.3.5. Comprehensive Measures

Newcomb's Modularity Index [17] (Equations (15)-(17)) is based on Design for Life Cycle, encompassing all aspects from initial conceptual design, through normal product use, to the eventual disposal of the product. The final Modularity value, $CR_{overall} \times CI$, is defined as the degree to which one lifecycle viewpoint's architecture corresponds to another lifecycle viewpoint's architecture. "Complete" modularity is said to be reached when there is one-to-one correspondence between the two architectures. Modular boundaries are sensitive to different viewpoints and the metric compares the closeness of modular boundaries from two different viewpoints.

$$CR_{ij} = \left| \frac{V_i(x) \cap V_j(x)}{V_i(x) \cup V_j(x)} \right| \tag{15}$$

$$CR_{overall} = \frac{\sum CR_{ij}}{\#modules} \tag{16}$$

$$CI = \frac{\#on\ block\ diagonal\ connections}{\#\ total\ connections} \tag{17}$$

All the output variables are ratios of same quantity or ratio of counted numbers. Hence they are unit less. Empirical studies were conducted on Chrysler LHS center console. This index, although more comprehensive, may be very difficult to implement in practice and may yield inconsistent results depending who is doing the analysis.

Kota [18] proposed a measure to capture the existing differences in product design strategies in comparing different manufacturers on their efforts towards standardizing components across models. The degree to which products in a family are differentiated is a marketing strategy decision, and thus beyond the scope of their metric. They included the following factors in their measure: The number of different types of components that can be ideally standardized across models (i.e., those that are non-product differentiating and non-influencers of conformance quality); Geometric features of components in terms of their sizes and shapes; Materials used across these components; Manufacturing processes that were used for their production; and assembly and fastening schemes used. PCI (Product Line Commonality Index) is defined in Equation (18).

$$PCI = \frac{\sum_{i=1}^{P} n_i \times f_{1i} \times f_{2i} \times f_{3i} - \sum_{i=1}^{P} \frac{1}{n_i^2}}{(P \times N) - \sum_{i=1}^{P} \frac{1}{n_i^2}} \quad (18)$$

N is the number of products in the product family, and P is the number of non-differentiating components that can potentially be standardized across models. f_{1i}, f_{2i}, and f_{3i} are multiplication factors for size, shape, manufacturing and assembly of each component and are assumed to be independent of one another .The PCI score is expressed as a percentage and takes values between 0 and 100. The size factor, f_{1i}, is computed as the ratio of the greatest number of models that share component *i* with identical size and shape to the greatest possible number of models that could have shared the component *i* with identical size and shape. The other two factors (f_{2i} and f_{3i}) are also computed in a similar manner.

While PCI does not consider market demand for different variants, Maupin's Weighted Simplicity Metric S (Equations (19) and (20)) does [19]. Indices proposed target Simplicity, Standardization, Direct Cost and Delayed Differentiation to help small manufacturers understand the flexibility of their process to engineer a product family. Q is the contribution of the model to total family sales, PN are the individual component and operations, n the number of components and operations required for a model, and m the total number of models in the family, and C is number of instances beyond the first of a component or process across the family.

$$S = \sum_{i=1}^{m} Q_i \sum_{j=1}^{n} PN_{j,i} \quad (19)$$

$$SI = \frac{\sum_{i=1}^{n} (C-1)_i}{S} \quad (20)$$

4.1.2.4. Requirements Adaptability

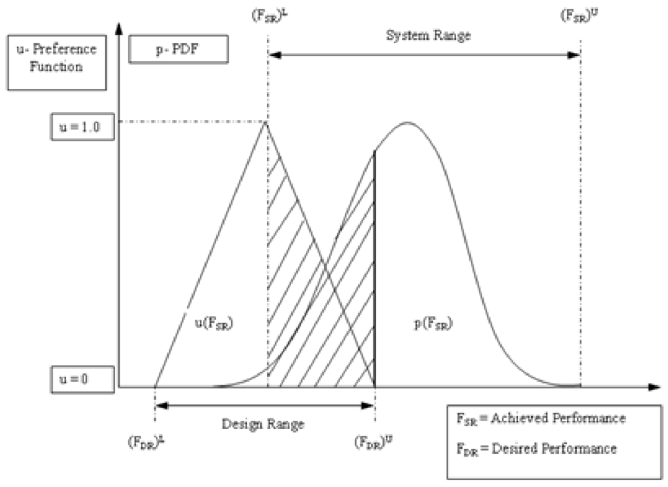
Finally, we discuss more general type of adaptability. The feasibility, difficulty, cost and time of design changes to meet unanticipated changes in requirements, supply chain, manufacturing, schedule, or financing. Changes could happen during product development, prototyping, full scale production or after full scale deployment. Requirements adaptability is dependent on the response to future unexpected changes. This is an area that has been least studied. It is also the area that perhaps offers the greatest potential for reducing product development cycles as envisioned in META.

Metrics for Redesign Adaptability can be sub-classified as those based on expected value, design variable sensitivity ratios, cost sensitivity and multi-factor measures that account for not only the cost of the change but the probability that such a change might happen. Forecasting such probabilities pose the most difficult challenge.

4.1.2.4.1. Metrics based on Expected value/Utility functions

To measure the goodness of fit of a design alternative with respect to the requirements, one compares its performance with respect to utility (preference) functions for each requirement.

Several metrics have been proposed along these lines. For example, Chen's Design Preference Index (DPI) (Figure 10) [20] is based on the expected value of preference functions of design performance within the range of design solutions.



$$DPI = E[u(F_{SR})] = \int_{F_{SR}^L}^{F_{SR}^U} u(F_{SR})p(F_{SR})dF_{SR}$$

$p(F_{SR})$ = probability density function of design performance

$u(F_{SR})$ = preference function of design performance

$E[u(F_{SR})]$ = Expected preference function value of achieved design performance over the range of solution

Figure 10 Chen's Design Performance Index

Jiao's Design Customizability Index [21] (Equation (21))) is a different form of the same idea. DPI is the design preference index.

$$CI^D = \frac{1}{1 - \log_2(DPI)} \tag{21}$$

It is not clear what advantage is gained by using this reciprocal log form, other than similarity to entropy measures for complexity. Values range from zero to one, where 0 implies ∞ information content and no customizability and 1 implies 0 information content and fully customizable design. In the form given, CI and DPI just measure the current fitness and not adaptability. However, one could look at them as the amount of flexibility available to meet a different range of requirements.

Another similar set are Simpson's Design Freedom (DF) and Information Certainty (IC) metrics [22] (Equations (22) and (23)). Instead of looking at the expected value of the preference function, they just compute the extent of overlap of the design parameter inside the target range (Similar to Suh's Information Content) Design freedom is the extent to which a system can be "adjusted" while still meeting its design requirements. In the equations, PR and TR are the feasible and target performance ranges.

$$DF = \frac{1}{n}\sum_{i=1}^{n}\frac{TR_i \cap PR_i}{PR_{i,initial}} \tag{22}$$

$$IC = \frac{1}{n}\sum_{i=1}^{n}\left(1 - \frac{(DV)Range_i}{Initial - (DV)Range_i}\right) \tag{23}$$

It appears that they consider the target range TR to be fixed and the performance range PR to be variable, i.e., the design is evolving. For an adaptability metric we might do the exact opposite.

4.1.2.4.2. Requirements sensitivity metrics

The most direct way to look at requirements adaptability is compute the sensitivity of design variable values to functional requirement values. Of course, these assume that just parametric variation would achieve the new requirement. Kalligeros [23] computes the sum of changes in all variables affected by a FR change by multiplying derivatives with delta FR, as shown in Equation (24).

$$\Delta x_i = \sum_{j=1}^{m} \frac{\partial x_i^s}{\partial FR_j^s} \Delta FR_j^s \quad (24)$$

A change can propagate to a variable directly because it is dependent on the change in functional requirement or indirectly in form of another variable that is directly dependent on the changing functional requirement. The units appear to be inconsistent since each x_i will have different units so addition would be incorrect.

4.1.2.4.2.1. Cost sensitivity metrics

A general adaptability metric looks at the cost sensitivity to change in requirements: (*Δ Cost /Δ FR*). However, this involves different units in the numerator and denominator. Therefore, it needs to be normalized. Shaw defined two adaptability metrics for satellites [24]. The first one measures the sensitivity of the cost to performance changes, defined as the ratio of 1% change in the Cost per Function (CPF) to 1% change in a relevant variable, *ΔCPF/CPF)/ΔX/X)*. The second metric measures the flexibility of an architecture for performing a modified mission as $(\Delta CPF/CPF)_x$,where *x* identifies a particular mission.

Hommes [8] "measures" cost in an indirect way, by assuming that the more variables that need to change, the greater the cost. They look at the depth of impact of change between parts or modules that interact. The concept of Change Cost (CC) is to measure the impact of a change in one module on the rest of the system, in terms of the percentage of other modules that are potentially affected. Change Cost is computed using the visibility matrix V (*aka* reachability matrix). CC is calculated as the ratio between the average of the sum of the rows in V and the number of elements. This appears to be a rather superficial metric since not all changes are equal in amount or cost.

4.1.2.4.3. Other measures

Rajan [25] applied an approach similar to FMEA to account for not just how much change occurs but the probability of its happening. This methodology is termed as Change Modes and Effect Analysis (CMEA). It is a two-step process: Decomposing the Product and Forming the CMEA table. A Change Potential Number (CPN) (Equation (25)) is calculated as a result that is equivalent of adaptability.

$$CPN = \frac{1}{N} \sum_{i=1}^{N} \frac{[(R_i + F_i) - O_i + 8]}{27} \quad (25)$$

The number “8” in the numerator and the number “27” in the denominator are used to bind the value of CPN between 0 and 1. The product decomposition can be done at Function, parts or modular level. As subjective ratings are involved, different people will have different values for CPN. Note that Flexibility (F), Readiness (R), and Probability of Occurrence (O) are all subjective values between 1 and 10.

4.2. Final Metrics

Table 4 Selected Complexity and Adaptability Measures

Measures	Participates in Peak Labor	Participates in Schedule	Participates in Complexity	Participates in Adaptability
Coupling	🟩	🟩		🟩
Cycles				
Nodes	🟩		🟩	🟩
Edges	🟩		🟩	🟩
Average Node Weight				
Average Link Weight				
Average Weighted Node Degree				
Maximum Weighted Node Degree				
Weighted Connectedness				
Weighted Information Content				
Normalized Weighted Information Content				
Average Weighted Clustering Coefficient				
Number of Nodes and Links				
Weighted Mobility				
Links		🟩		🟩
Average Node Degree		🟩		
Maximum Node Degree	🟩			🟩
Connectedness		🟩	🟩	
Information Content	🟩	🟩		
Normalized Information Content		🟩	🟩	
Average Clustering Coefficient	🟩	🟩	🟩	🟩
Node Weight + Link Weight				
Mobility	🟩			🟩
Total Node Weight				
Total Link Weight				
Modularity		🟩		🟩
Integration Complexity				
System Modularity				

complexity and adaptability. From this menagerie we selected a set of 28 for implementation in the prototype complexity and adaptability tool and that would then be used in experimentation and calibration (Table 4 – repeated here). The table also indicates which measures appeared in any of the correlated peak labor, complexity and adaptability metric functions produced during calibration, which will be discussed in the next section.

There were four main factors which drove the selection of these 28 measures. Do the measures reflect our expectations of what system complexity and adaptability should be? Are the measures generally applicable? Are the measures objectively quantifiable? Can the measures be computed from artifacts that will be available early enough in the META design process to support design choices?

The first factor covers the reasonableness of the measures. In some sense, all of the candidate measures discussed in the previous section pass this test, since we only considered candidates that had been reasonably proposed to measure some aspect of complexity or adaptability. This was confirmed by our initial data collection activities (see the next section) in which we worked with engineers from various Boeing programs to identify those elements which they believed contributed to system adaptability and complexity. One of the most frequent answers was the number and type of interactions that were present between system components. This was not unexpected, but served to confirm the utility of many of the graph and network based metrics.

The second factor is the generality of the measures. Our objective is to develop a set of measures and metrics that can be used by the META design flow and process, and integrated into the META tool chain, and that can be applied to any CPS that META is used to design. Accordingly, we sought measures of broad applicability. This led us away from domain and industry specific measures.

The third factor is related to the second, we sought measures and metrics that could be objectively quantifiable and were effectively computable. This was to avoid the outcome being dominated by subjective judgments or qualitative factors, and to make it possible to integrate the metrics into large scale design space exploration to support early design selection decisions. In order to use the metrics to support decision making (either explicit decisions by human stakeholders, or implicit decisions made using multi-objective value functions), it must be possible to rapidly and automatically calculate the measures and metrics. Even in the comparatively small space of 522 viable designs of the IFV challenge problem discussed at the end of section 4.5.2, measures and metrics that require subjective evaluations would be overwhelmed. These considerations ruled out a number of the candidate metrics.

The final factor considered is whether or not the measures and metrics can be calculate from artifacts that will be available as part of the META design process. In particular, since we believe that in order to support the META goals of a 5x reduction in program schedule and cost, the metrics must support the selection of preferred designs from a set of candidates in the early stages of the design and development process. Accordingly, the metrics must be applicable to those artifacts available at that time that could potentially distinguish candidate designs. The most likely artifacts to be available, and to which the metrics should be applicable, are system architectures, including logical, physical and functional. In common with many design representations, such system architectures can be represented as graphs, so we focused on metrics that can be formulated in terms of graph structures. Note that that is not that severe a restriction, since in fact most of the initial candidate metrics discussed above either are already defined in terms of graphs, and in general architecture graphs, or can be reformulated to apply to such graphs.

These considerations led to the selection of the majority of the measures listed in Table 4. Some of the measures were included however because they were either easily calculated given the other measures present, or are a byproduct of calculating other measures.

4.3. Calibration

The previous sections describe algorithms to compute metrics for design complexity and adaptability. These metrics need to be verified and calibrated with historical Boeing design data and expert judgment. In this section we define the META calibration problem and describe statistical methods and possible challenges associated with calibration. For simplicity, the discussion is generally framed in terms of complexity metrics, but the same calibration approach is being used for both complexity and adaptability metrics. The section begins with defining the overall calibration approach and methodology, then discusses the data being collected to support calibration, and finally includes the results of calibrating the metrics.

The data being collected consist of < observable, design artifact> pairs, where the observables are the program attributes that the measures are being used to estimate. In this case, they consist of peak labor (as a portable cost metric) and schedule. We also collected expert assessments of system complexity and adaptability to provide insights into how well the measures correlate with intuitive notions of complexity and adaptability (since this work is attempting to define objective complexity and adaptability measures there are no accepted historical measures of these two quantities to serve as baselines). We then transform the design artifact element of each pair into the graph representation used by our metric tools to calculate the complexity and adaptability measures. The measures are then calculated for the design artifacts, resulting in a set of <observable, <metric vector>> pairs. These pairs are then analyzed for correlations to derive functions that serve as estimators for the observables. These estimators are then the calibrated metrics.

4.3.1. Approach

Design representations typically capture multiple design views (e.g. requirements, functions, logical, physical, manufacture, etc.) at multiple levels of abstraction. Let us start by considering a single level of abstraction. Let $\boldsymbol{x} \in \Re^n$ denote a vector containing the attributes of a design at a particular level of abstraction. Let $C_d^l(x), d = 1,...,m; l \in \{a_1,...,a_p\}$ be a set of scalar complexity measures of the design. The subscript *d* corresponds to the particular component of complexity, such as functional complexity, manufacturing process complexity, and design process complexity, etc. We assume that we have *m* such components. These components could well correspond to individual design views. The superscript *l* corresponds to the various computed metrics $\{a_1,...,a_p\}$. The computed metrics use $\boldsymbol{x}$ as input. Since the $C_d^l(x)$ values are obtained from known implementations, we assume that the functions $C_d^l(x)$ are known (not necessarily analytically, but implicitly). It is unclear whether $C_d^l(x), d = 1,..., m$ are independent of each other. In fact, it is quite likely that they are not. We express the overall complexity of a design as follows:

$$C(x) = f(C_d^l(x), d = 1, ..., m; l = 1, ... p) \tag{26}$$

where the true functional form of $\boldsymbol{f}$ is unknown. However, we can approximate $\boldsymbol{f}$ by a function $\boldsymbol{g}$ whose functional form is known or can be determined from the data. This leads to the estimation problem of

$$C(x) = g(C_d^l(x), d = 1, ..., m; l = 1, ..., p; \theta, \varepsilon) \tag{27}$$

where the parameters $\boldsymbol{\theta}$ can be estimated from the data and $\boldsymbol{\varepsilon}$ is the error in estimation.

We state our calibration problem as one of finding function $\boldsymbol{g}$. Our calibration method will need to consider the properties that we seek from our complexity metric *C*. Currently, we consider the following properties:

1. ***C*** should correlate to the impact on schedule of complexity
2. ***C*** should be a relatively smooth function of ***x***. This would support having ***C*** as a design objective that we would seek to optimize during design optimization.
3. ***C*** should be self consistent across the levels of design abstraction, i.e. as ***x*** changes, going from an abstract to a detailed representation of a design, we expect that ***C*** is somewhat consistent across levels.

4. ***C*** should have an associated uncertainty which decreases as we move from abstract to detailed design.

The primary source of calibration data, since we are seeking to establish a correspondence between complexity and schedule, is the actual (and potentially predicted) schedule of the projects being used for calibration. An additional source of calibration data is the opinion of experts on complexity. This expert judgment can be used both to provide direct "measures" of complexity, so that we can examine our metrics not just against their predictive power with respect to schedule, but also to their "accuracy" in measuring perceived complexity. With respect to expert assessments of complexity should we assume that expert opinion for a product can be gathered along the lines of the individual components $C_d(x)$ or should we consider opinion on the overall complexity ***C***? Should we assume that expert opinion can be gathered for each abstraction level ***x***, or is the expert opinion available at only the most detailed level of design representation?

It could be argued that for an ongoing design effort, expert opinion that is limited to a particular level of abstraction could be gathered along the way. However, for a historical product, i.e. one whose design has been completed, it would be hard for experts to express opinions that are limited to a particular level of abstraction.

Without fully rationalizing the availability of valid expert opinion, let $C_d^k(x), d = 1,...,m; k \in \{e_1,...,e_q\}$ be a set of complexity values obtained. As before, the subscript *d* corresponds to the particular component of complexity. Also, like before, the superscript corresponds to the source of the data, in this case, the particular expert. We assume that we have ***q*** data sources. Table 5 captures all these complexity values for a single product. In addition to the metrics described above, two additional set of metrics on Table 5 need explaining. The bottom row of the table describes a set of overall complexity measures that aggregate the measures along their respective columns. The rightmost row describes the true value of complexity components, which would represent an aggregation of values within their respective rows. This true complexity value is of course unknown to us.

Table 5 Complexity values for a particular product

Complexity Component (d)	Source												
	Computed								Experts				Truth
	Coupling	Cluster Coeff	Avg Node Degree	$N\ l_{vd}$	Mobility	Connect-edness	. . .	p	1	2	. . .	q	
Requirements Complexity													C_{d_1}
System Logical complexity													C_{d_2}
. . .	Complexity, as calculated from network measures								Complexity, as obtained from experts				
Component m													C_{d_m}
Overall Complexity	C^{a_1}					C^{a_2}		C^{a_p}	C^{e_1}	C^{e_2}		C^{e_q}	C

We next consider multiple products. Let us assume that we gather schedule data and/or expert opinion for each of $\boldsymbol{n}$ historical products (at their most detailed level of abstraction). We assume that values $C_d^k(x), \forall k \in \{e_1,\ldots,e_q\}$ for any given product will vary. Let $\boldsymbol{x}_j$ denote the design attributes for product $\boldsymbol{j}$ where $\boldsymbol{j}$ = 1, …, $\boldsymbol{r}$. Source e_k evaluates product $\boldsymbol{j}$ along the lines of each of the complexity components d and assigns it a schedule/complexity value $C_d{}^k(x_j)$. However, the confidence in this value could vary by component. Let $0 \le w^{jdk} \le 1$ denote this confidence where a value of 0 means no confidence and the value of 1 means absolute confidence.

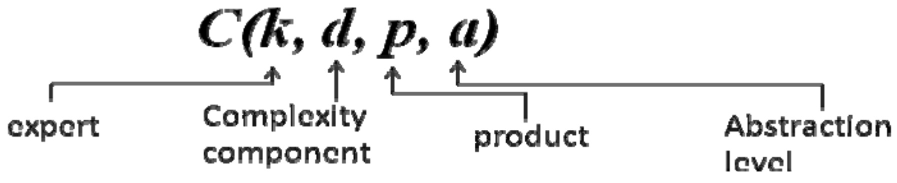
Consider, for a moment, the overall complexity $C^k(x), \forall k \in \{e_1,\ldots,e_q\}$ data source. For each product, $\boldsymbol{j}$, we can then calculate the "true" complexity of that product as a weighted average of the values over all sources:

$$\bar{C}(x_j) = \frac{\sum_k w^{jk} C^k(x_j)}{\sum_k w^{jk}} \tag{28}$$

Hence, our calibration problem can now be written as estimating the best fit for the function $\boldsymbol{g}$ such that

$$\bar{C}(x_j) = g\left(C_d^i(x_j), d = 1,\ldots,m; i = 1,\ldots,p; \theta, \varepsilon\right);\ j = 1,\ldots,r \tag{29}$$

Overall, we consider the complexity measure to be depending on primarily four factors: the source of the measure (expert opinion, computed, actual/predicted schedule), the complexity component, the product and the level of abstraction. This is indicated below.



Collecting actual and projected schedule data involves matching schedule data (from integrated master schedules or project plans or other sources) to the design artifacts. To collect the data from experts to get a direct assessment of complexity to supplement the correlation with schedule, we use the following approach:

1. Collect information on the different pieces of $\boldsymbol{x}_j$ for several products. The elements of $\boldsymbol{x}_j$ will contain information such as price, schedule, number of designs etc that contribute to complexity
2. Create an information sheet for each product (possibly sub-systems within a larger aerospace product) that will clearly define the boundary of the sub-system. This boundary will align with the boundary used to obtain $\boldsymbol{x}_j$. It will also include a description of the information in $\boldsymbol{x}_j$.
3. We ask experts to describe the source of complexity as they perceive it, as well as ask them to assign a quantitative value of complexity from 0 to 100.
4. We seek the experts' estimate of design time and cost.
5. We seek the expert's level of familiarity with the product. This will help us in creating the weights that we will assign to each expert's complexity value.

4.3.1.1. Errors in Estimation

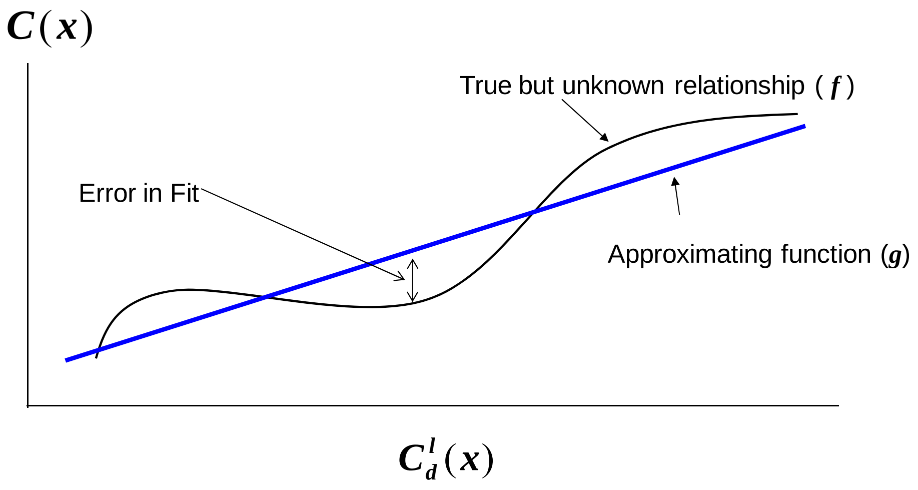
Since we are estimating an unknown true function, $\boldsymbol{f}$, using schedule data and engineering judgment and approximating functions, there are two errors that we have to deal with. These are

1. Error in fit
2. Noise in the data

For all the figures given below, we only show a notional picture for a single component (*d*) and for a single product (*l*). The horizontal axis represents the metric calculations based on the algorithms and the vertical axis represents the "true" value of the metric.

1. Error in Fit

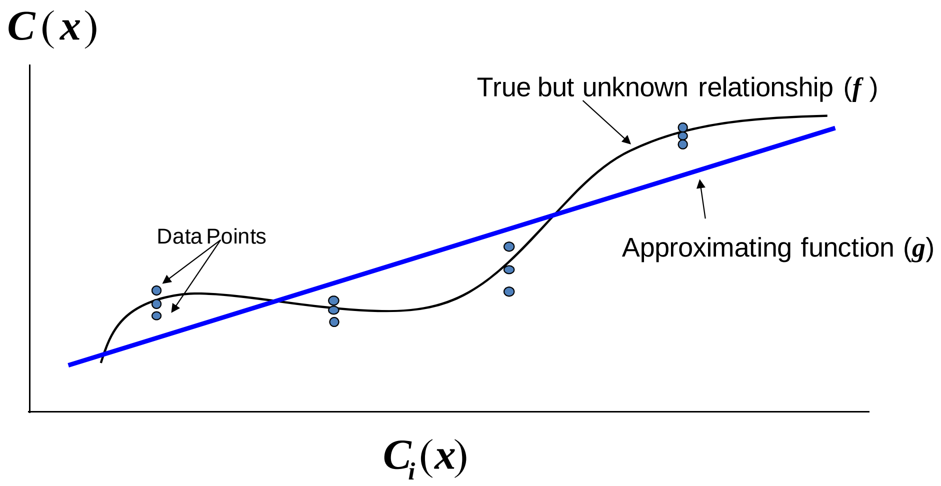
Consider Figure 11.



$C(x)$
True but unknown relationship (*f*)
Error in Fit
Approximating function (*g*)
$C_d^l(x)$

Figure 11 Error in Fitting the Approximation Function

In Figure 11, the unknown true function is represented using the black line and the blue line represents the model fit based on the data. The error in the fit, also called lack of fit, is the



$C(x)$
True but unknown relationship (*f*)
Data Points
Approximating function (*g*)
$C_i(x)$

Figure 12 Error Due to Data Noise

difference between ***f*** and ***g***.

2. Noise in the data

Consider Figure 12.

In Figure 12, for a given ***x***, each dot represents the value of the complexity from an expert, or corresponding schedule data. Although $C_i(x)$ is same for the given value of ***x***, the actual schedule derived from the data can vary due to effects independent of complexity, and the complexity-influenced "true" schedule could vary from the reported value. Likewise, expert opinions can vary and could lead to different reported values of complexity due to biases of the expert or other factors (hazy recall, etc.). However, the expectation is that these data values

Table 6 Sources for Calibration Data

Source	Program	System/Sub-systems gathered
BCA	787	ATA 42(Integrated Modular Avionics), ATA 22(Auto Flight), ATA 23 (Communications), ATA 24(Electrical Power), ATA 27(Flight Controls), ATA 29(Hydraulics), ATA 30(Ice and Rain Protection),ATA 31(Indicating /Recording System), ATA 33(Lights), ATA 34(Navigation), ATA 46(Information System)
BDS	F/A-18F	Air Vehicle Targeting FLIR Pod Subsystem
BDS	EA-18G	Air Vehicle A/C Modification Kit (Changes to F/A-18F to make EA-18G)
BDS	AV-8B	Air Vehicle
BDS	C-17	Air Vehicle (Large Transport)
BDS	F-15	Air Vehicle, Targeting FLIR/Laser Pod Subsystem Radar Subsystem
BDS	QF-16	A/C Modification Kit (Changes to F-16 to make QF-16)
BDS	J-UCAS	Air Vehicle (UAV)
BDS	Laser JDAM	Smart Weapon
BDS	Small Diameter Bomb	Smart Weapon
BDS	P-8A	Mission Subsystem, Avionics Subsystem

(schedule, judgment) will vary around the true complexity value. This variation is characterized as the “data noise error”.

4.3.2. Data

We have collected historical design data from Boeing Commercial Aircraft and Boeing Defense Space, and Security. We believe that this diversity of commercial and military systems gives us a reasonable representation of products from the aerospace domain. In both cases, we sought sub-systems that had a good mix of cyber and physical elements. Table 6 lists the programs and sub-systems that we used to perform our calibration.

For BCA, we decided to examine different systems and sub-systems on our most recent commercial airplane, the Boeing 787. We used the system logical representation of the 787 from a modern Product Data Management PDM system; this representation essentially is a schematic-level description of cyber-physical systems. The raw data amounts to 3.1GB of XML, representing 2.2 million elements and links between them.

The design of the Boeing 787 started in 2002 using the latest CAD/PDM/PLM tools available at that time. The company built a new tool to support the System Engineering effort during preliminary design. This tool employed a rich data model that was developed to meet the technical data requirements of the System Engineering effort, as well as to support the large supplier involvement on the program. One of the elements of this data model was a logical representation of the systems on the airplane. This data captures both the physical and the cyber aspects of how system functions are implemented, at a level of abstraction similar to that of an engineering schematic diagram. It also contains, at a high level, a hierarchical organization of the system components.

The 787 system logical representation contains information about various parts, assemblies and their instances as well as their connections without any physical (or geometric) representation. For example, it would provide information about a battery (e.g., manufacturer, capacity, voltage), in which assemblies this battery is instanced and what is connected to the battery.

The data is organized as objects (in the sense of object oriented programming) using 112 different classes. These classes can be categorized into several groups:

- Parts and assemblies - These are objects that can be instanced in (other) assemblies
- Instances - Instances of parts or assemblies used within an assembly

- Ports - Ports are points of connectivity on a part. Link objects are used specify that a port (of one part instances) is connected to another port (usually on a different part instance).
- Parameters, Signals - Parameters provide additional attributes for other objects. Link objects are used to associate parameters to the object to which the parameter applies.
- Links - Links are a generic mechanism to relate two objects. See appendix 2 for a complete list of link types.
- Supplemental - Classes that are used only as complex attributes of other classes
- Miscellaneous - other objects used for internal organization etc.

Each of the 112 classes defines its own set of attributes. Commonality between these attributes suggests that some inheritance relationships could be defined between the classes but the classes were clearly not defined with an extensive inheritance hierarchy in mind.

For BDS, we examined systems and sub-systems from a variety of military programs, some of which had been initially designed almost three decades ago, with modifications and upgrades over the years. As can be imagined, the extent of computer-represented data varied significantly for the BDS data. Therefore, alternative data sources were required. We obtained technical product documents such as schematic diagrams, WBS reports, product knowledge from design engineers, etc To make things uniform, we used data from system technical documents to create logical representations at an abstraction level was either similar to or one level higher than the BCA data. We then used the system logical data to build the abstract weighed graph required by the metrics algorithms.

The most common forms of design description for the BDS programs were:

- A Work Breakdown Structure (WBS) Component Description Report for lists of system components (nodes);
- System schematic diagrams from the programs for defining interactions;
- System specification, interface documents, and design reviews for more detailed functional information including data bus and electrical power loading;
- Engineering judgment on design relationships among the components/nodes – physical/assembly, electrical/cooling/hydraulic (energy), informational, matter, and actuation/force – and the strengths of the relationships.

The WBS Component Lists offered a hierarchical, logical model of the system within the system boundary that we chose to model comparable types of systems. System schematic diagrams shows how the subsystems are connected to others for information interaction via various multiplex busses, and serial and discrete lines.

We consulted a large number of sources within the company to gather the observables (schedule, peak labor, complexity, adaptability, reliability). This includes objective data about a product's cost, design time, and reliability as well as engineering judgment on observables such as complexity and adaptability for which there are no (currently) established objective metrics. Unfortunately, the reliability data that we obtained was not suitable for calibration, since all of the systems being used for calibration were flight/safety/mission critical systems with extremely high reliability rates, resulting in an insufficient spread of values to support calibration.

To gather this data, we interviewed lead engineers for the systems/sub-systems. For the objective observables, we chose terms of reference that would be consistent across BCA and BDS systems. For the subjective quantities, we needed to establish a common scale for complexity and adaptability, and seek responses that were normalized with each other. We defined complexity and adaptability values as being in the interval [1,100]. Next we chose a system that was familiar to all lead engineers, and provided the complexity and adaptability values obtained from its lead engineer to all others as a benchmark. The leads were asked to provide their complexity and adaptability values relative to the benchmark.

4.3.3. Calibration Results

Once the design artifacts were collected for each of the programs being used for calibration, they were used to create weighted directed graphs representing the product architecture to which the measures could be applied. We calculated values for 24 of the 28 metrics in Table 4 for the BCA products, and 26 of 28 for the BDS products. The discrepancies were due to technical issues with the graph structures for some of the products and certain of the metrics. Table 7 provides a sample of the metrics values computed for the chosen products. Historical data provided a total of 11 external observables that characterized the design process for these products. The 11 observables include: Planned-design-time-to-Gate10, planned-design-time-to-Gate11, Realized-design-time-to-Gate10/11, Start-Labor, Peak-Labor, Peak-

Table 7 Sample Measure Values

	Average Node Weight	Average Link Weight	Weighted Average Node Degree	Weighted Maximum Node Degree	Weighted Connectedness	Weighted Information Content	Weighted Normalized Info. Content	Weighted Average Clustering Coeff.	Overall Weight	Weighted Mobility
ATA_23	15250.4	1	17.1596	233	0.023	5.59E+09	111.234	0.073464	3602	5742680
ATA_23	5695.44	1	4.10938	233	0.008	1.33E+09	59.8649	0.089208	782	1458840
ATA_26	6321.41	1	6.0462	223	0.01	1.49E+09	46.9229	0.088376	1219	1917220
ATA_27	67.4465	1	5.2	143	0.006	11155000	0.167045	0.020379	1548	31063
ATA_29	14416.5	1	5.39084	231	0.007	4.83E+09	98.8828	0.057668	1371	5350420
ATA_31	10690.6	1	9.53846	247	0.017	2.09E+09	74.494	0.008307	1650	3060740

Supplier-Labor, Reliability, Start-Complexity, Final-Complexity, and Final-Adaptability.

We solve the estimation problem by:

1. Scaling the computed measures so they have approximately the same range of values
2. Defining a number of non-linear terms involving the computed metrics
3. Constructing parametric expressions involving one, two or three terms, obtained from a mix of linear and non-linear terms. An intercept term was also added. Two types of non-linear terms were constructed: products of metric pairs, and log values of metrics.
4. Computing parametric values corresponding to a least squared error fit between the expressions constructed in item 3 above and the observables.

The above calibration was performed in two ways using the BCA and BDS data separately. We calibrated four of the observables listed above, i.e.:

1. Schedule (Realized-design-time-to-Gate10/11),
2. Cost (Peak-Labor),
3. Complexity (Final-Complexity), and
4. Adaptability (Final-Adaptability).

Our objective was to investigate whether any of the computed metrics, individually considered or combinations thereof, are good predictors of the design observables. If so, they would serve as a metric for complexity or adaptability, for use in preliminary design stages to estimate a design's future cost, schedule, and reliability.

To indicate the accuracy of fit of the estimating functions to the data from the chosen products, we provide plots showing the estimated values vs. values from historical data for each of the designs in Table 6. We also provide RMS-error and R-squared for each of the calibration models. The following section provides results from our calibration study (Figure 13, Figure 14).

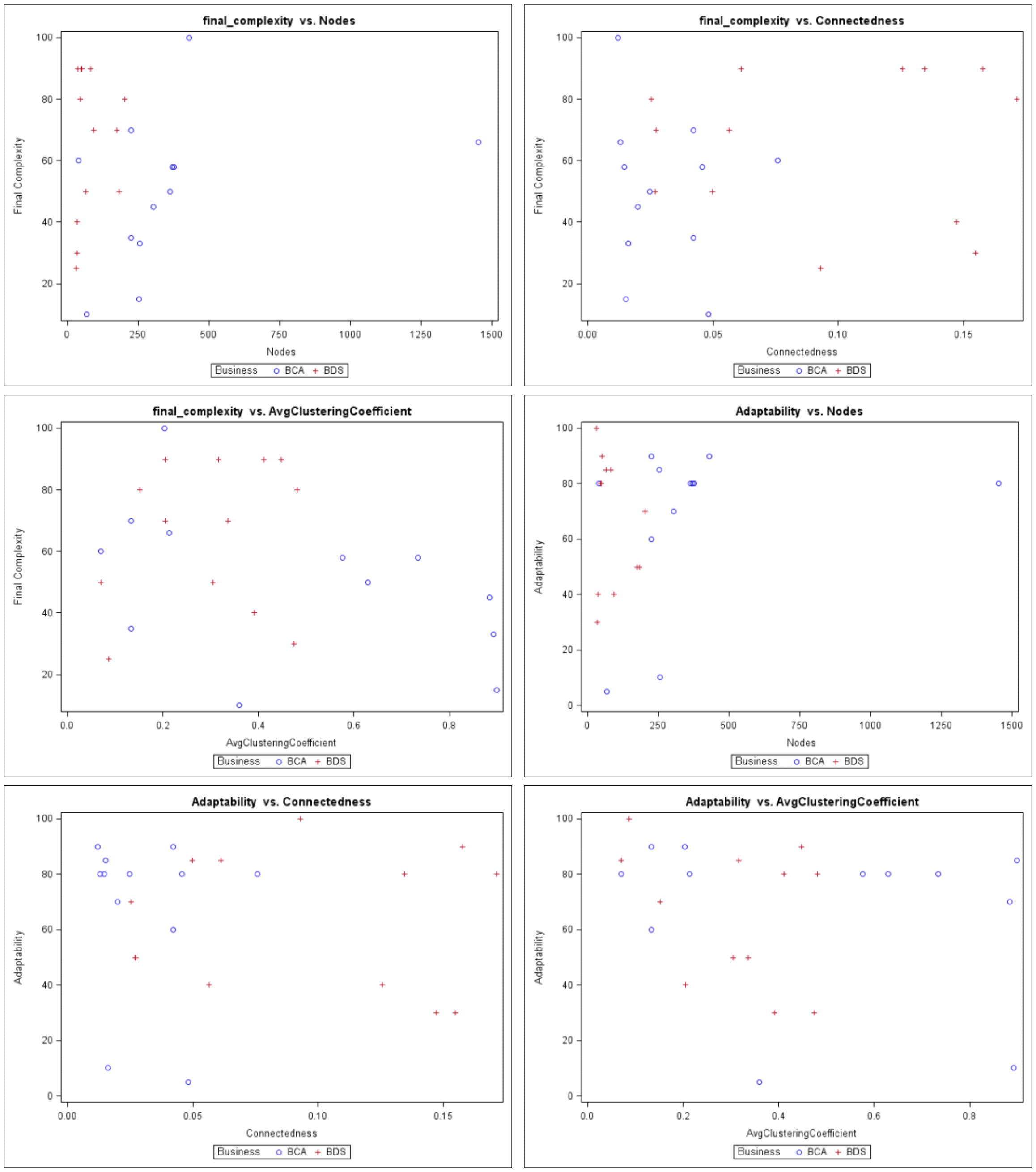


Figure 13 Plots of Complexity and Adaptability vs. NodeCount, AvgClusteringCoefficient, and Connectedness for Both BCA and BDS Datasets

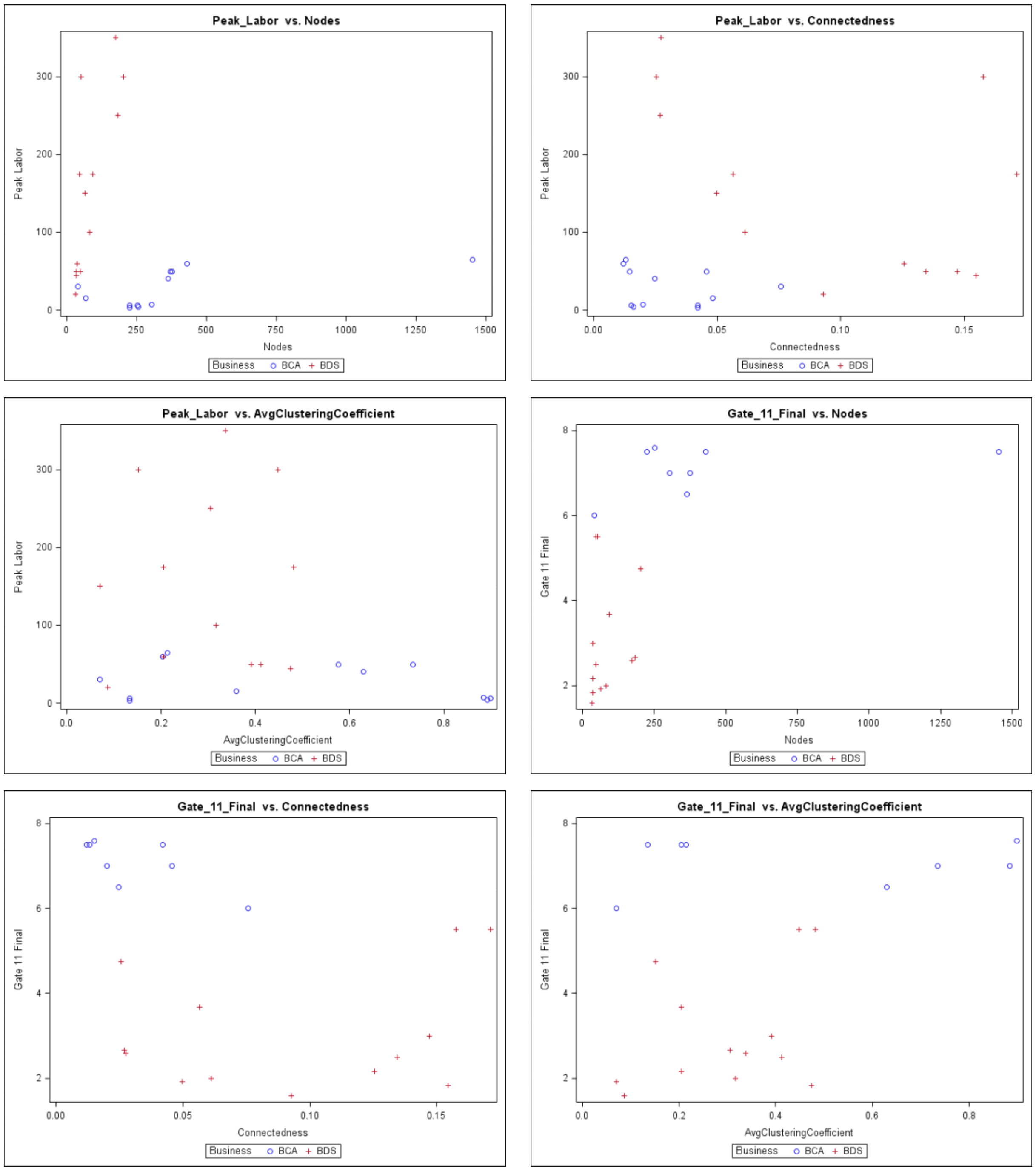


Figure 14 Plots of Peak Labor and Schedule vs. NodeCount, AvgClusteringCoefficient, and Connectedness for Both BCA and BDS Datasets.

Note that our sample size in each of the calibration exercises involving BDS and BCA data separately, is about twelve to fifteen. Although this is a reasonable size to provide an initial examination of the predictive capability of our META metrics, one should avoid the pitfall of making strong inferences from these initial results. A far larger study would need to be performed, involving more products and additional design representations, to make stronger claims.

The scatter plots do not indicate a clear relationship between the observables and the

Table 8 Calibrated Metrics – BDS Data

BDS Data Only					
		Terms in Model	RMSE	Adjusted R-Squared	Preferred Model
Complexity	1 - Term	14.2324 + 5.10425*LN(AvgNodeDegree) * LN(Mobility)	20.26	0.29	
	2-Terms	3.9754 - 2.717 * LN(Coupling) * LN(MaxNodeDegree) + 13.232 * LN(Links) * LN(AvgNodeDegree)	19.59	0.33	
	3-Terms	105.4001 +13.1307*LN(AvgNodeDegree) * LN(Mobility) + 22.5622 * LN(MaxNodeDegree) * LN(AvgClusteringCoefficient) - 651.395 * AvgClusteringCoefficient * AvgClusteringCoefficient	14.14	0.65	✓
Adaptability	1 - Term	37.1129 - 3.5749 * LN(Coupling) * LN(AvgClusteringCoefficient)	22.67	0.15	
	2-Terms	45.5896 + 10.9609 * LN(Coupling) * LN(AvgClusteringCoefficient) - 14.1096 * LN(AvgClusteringCoefficient) * LN(Mobility)	19.13	0.39	
	3-Terms	100.5166 + 16.5309 * LN(Coupling) * LN(AvgClusteringCoefficient) -16.0493 *LN(AvgClusteringCoefficient) * LN(Mobility) - 209.85 * AvgClusteringCoefficient*Modularity	17.30	0.51	✓
Peak Labor	1 - Term	-617.6944 + 145.69 * LN(Edge)	55.95	0.76	
	2-Terms	-773.25 + 234.0793 * LN(Edge) - 91.1 * LN(MaxNodeDegree)	47.13	0.83	
	3-Terms	-425.3753 - 12.0086 * LN(Coupling) * LN(Nodes) + 42.8054 * LN(Edges) * LN(IvdDisp) - 34.6139 * LN(MaxNodeDegree) * LN(Mobility)	36.13	0.90	✓
Schedule (Gate 11)	1 - Term	0.3082* LN(Coupling) * LN(AvgNodeDegree)	0.67	0.96	
	2-Terms	0.3753* LN(Coupling) * LN(AvgNodeDegree) - 5.677 * AvgClusteringCoefficient * Modularity	0.59	0.97	✓
	3-Terms (No Intercept)	0.3318* LN(Coupling) * LN(AvgNodeDegree) - 11.67 * AvgClusteringCoefficient * Modularity + 0.5395 * AvgNodeDegree * Modularity	0.55	0.97	
	3-Terms	0.6887 + 21.486* AvgClusteringCoefficient - 5.56 * AvgNodeDegree * AvgClusteringCoefficient + 0.11091 * IvdDisp * NormalizedInfoContent	0.37	0.93	

calculated metrics. This holds true for all the metrics, although for space considerations they are not all shown in this report.

As a next step, we define additional candidate terms — quadratic terms, cross products or interaction terms, and logarithmic terms. Logarithmic terms were chosen to account for the differing scales of the metrics and the historical data. We use these additional terms to build linear calibration models, where the number of terms was varied between one and three. Additionally, an optional intercept term was included.

Table 9 Calibrated Metrics – BCA Data

BCA Data Only

		Terms in Model	RMSE	Adjusted R-Squared	Preferred Model
Complexity	1 - Term	32.6014 + 4.8761 * LN(Connectedness) * LN(AvgClusteringCoefficient)	21.51	0.25	
	2-Terms	-64.06 + 1.8521 * LN(Nodes) * LN(Edges) + 8.1107 * LN(NormalizedInformationContent) * LN(AvgClusteringCoefficient)	8.25	0.90	
	3-Terms	-94.895 + 2.4668 * LN(Nodes) * LN(Edges) - 33.2453 * LN(Connectedness) * LN(AverageClusteringCoefficient) + 36.8216 * LN(NormalizedInformationContent) * LN (AvgClusteringCoefficient)	4.84	0.96	✓
Adaptability	1 - Term	28.1197 + 1.1256 * LN(Links) * LN(MaxNodeDegree)	27.85	0.09	
	2-Terms	-44.389 - 22.0488 * LN(AvgClusteringCoefficient) + 1.9716 * LN(Node) * LN(Edges)	26.09	0.23	
	3-Terms	-97.232 - 93.2936 * LN(AvgClusteringCoefficient) + 3.0922 * LN(Nodes) * LN(Edges) + 9.5168 * LN(Edges) * LN(AvgClusteringCoefficient)	24.89	0.30	✓
Peak Labor	1 - Term	13.2285 + 0.0306 * Nodes	19.75	0.32	
	2-Terms	18.4768 - 71.303 * AvgClusteringCoefficient * AvgClusteringCoefficient + 0.20381* Nodes * AvgClusteringCoefficient	15.63	0.57	
	3-Terms	20.1298 -140.873 * AvgClusteringCoefficient * AvgClusteringCoefficient + 0.47103 * Nodes * AvgClusteringCoefficient - 0.02673 * Nodes * MaxNodeDegree	12.44	0.73	✓
Schedule (Gate 10)	1 - Term	-1.1689* LN(NormalizedInfoContent)	1.26	0.94	
	2-Terms	660.29 * NormalizedInfoContent * AvgClusteringCoefficient - 0.2332 * LN(Links) * LN(AvgClusteringCoefficient)	1.12	0.95	
	3-Terms	879.88* NormalizedInfoContent * AvgClusteringCoefficient - 1.9065 * LN(AvgNodeDegree) * LN(AvgClusteringCoefficient) - 17.78 * AvgNodeDegree * Connectedness	0.66	0.98	✓

For each set of one, two and three term expressions, we determine the best model (expression) using the R-squared criteria. Additionally, we consider the statistical significance of the coefficients and the RMS error to decide which of the three models is most suitable for our use. This is done for the calibration problem for each observable. Table 8 and Table 9 show final calibration functions for four design observables, namely complexity, adaptability, cost and schedule in terms of the computed metrics for BDS and BCA data analyzed separately.

To visually evaluate the quality of the calibration models, we plot the observed vs. predicted (from model) data for the preferred models shown in the above tables. These are shown as scatter plots in Figure 15 and Figure 16. Data points along the diagonal indicate good

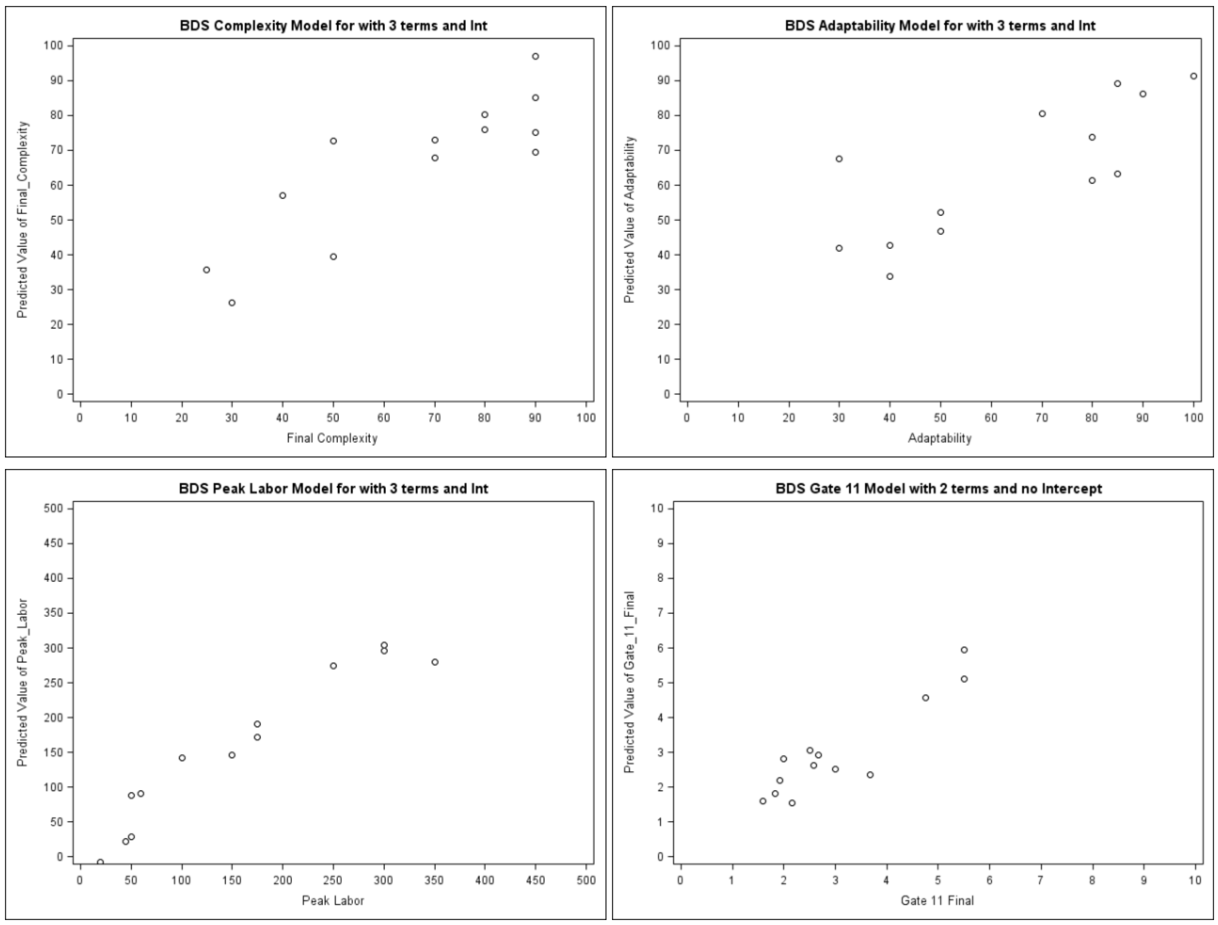


Figure 15 Plots of Complexity and Adaptability vs. NodeCount, AvgClusteringCoefficient, and Connectedness for Both BCA and BDS Datasets

quality fits.

As can be observed from the scatter plots, we have obtained reasonable calibration results for all except BCA adaptability.

In addition to enabling us to define the overall calibration functions for peak labor and schedule with credible degrees of correlation, our calibration activities have prompted a number of observations.

1. Product terms tend to dominate the estimating functions. This indicates that the metrics individually aren't good predictors of the four design observables, but rather combinations of them are.
2. Based on the RMSE and R-squared error values in the Tables, we conclude that our META metrics tend to not only be good indicators of complexity and adaptability, but are also reasonable predictors of Cost (Labor requirements) and Schedule (Design time).
3. Some metrics, such as AvgClusteringCoefficient, Mobility and Coupling , show up in multiple calibration models. It may be concluded that these are better

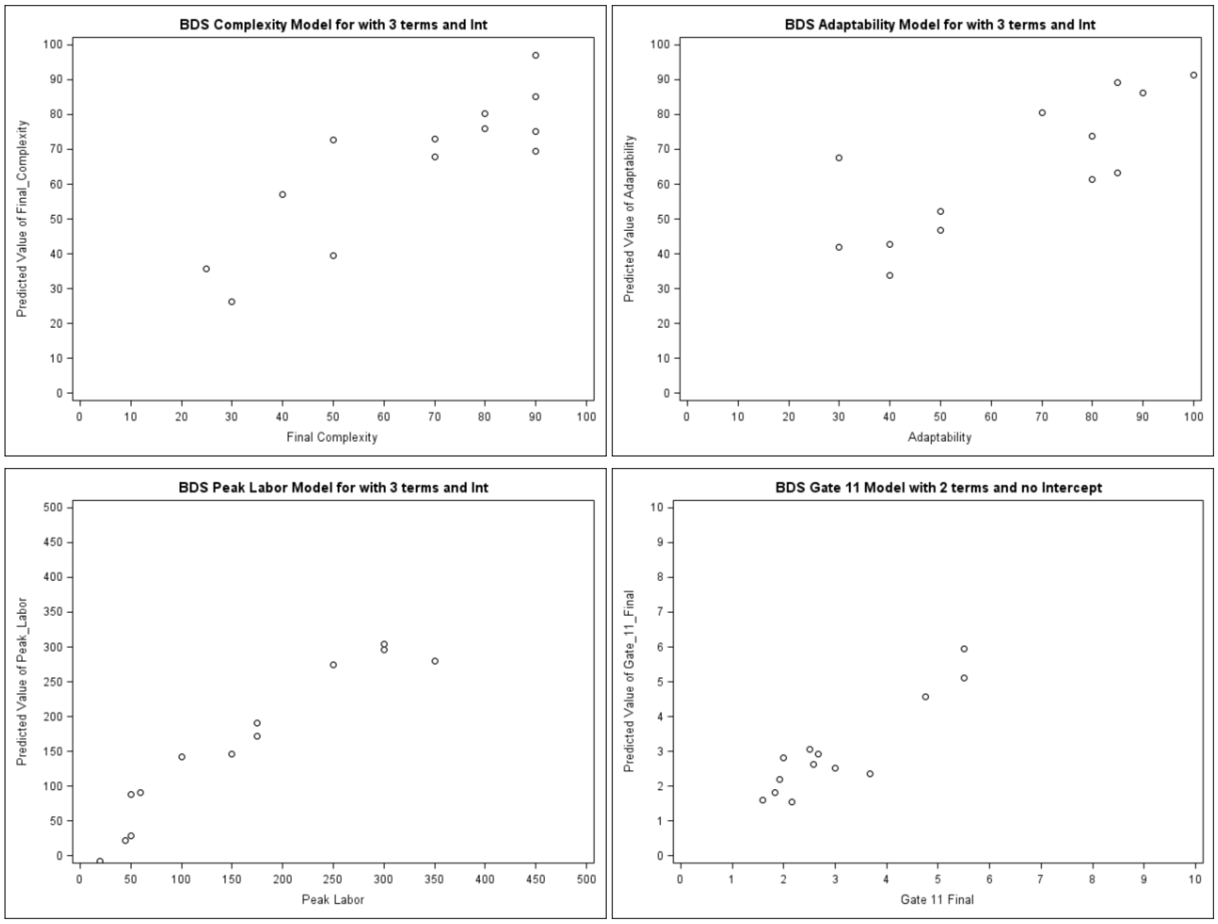


Figure 16 Plots of Complexity and Adaptability vs. NodeCount, AvgClusteringCoefficient, and Connectedness for Both BCA and BDS Datasets

complexity and adaptability metrics than the others. It would be interesting to examine these and refine them further.
4. Many of the terms appearing in the estimating functions are logarithmic functions of the META metrics. This means that the metrics are measuring attributes that are growing exponentially with traditional indicators of complexity such as size. This makes sense since many of our proposed META metrics are network

measures, such as path counts, which indeed tend to grow exponentially with the number of nodes.

4.4. Tool Prototype

The Complexity and Adaptability Metrics Tool Prototype was designed to use the weighted graph representation of candidate systems in the META design exploration process and compute the resultant calibrated complexity and adaptability metrics of the systems in consideration. To support the metrics analysis, the tool provides import capabilities, graph normalization and aggregation and the application of multiple metrics analysis algorithms analysis to derive the calibrated metrics. The tool supports integration with META design tools such as GME/CyPhyML as well as providing for direct input of artifacts to be analyzed. The tool has an open architecture to support future evolution and extension, including support for new input sources, new metric algorithms, new calibration results, and new metric consumers. Given the commonality of many of these features, and indeed some of the measures, a single tool prototype was created that implements both complexity and adaptability metrics.

The tool consists of three main elements that implement the core functionality of the tool (Figure 17). The central element is the graph store component. A SQL database is used to store

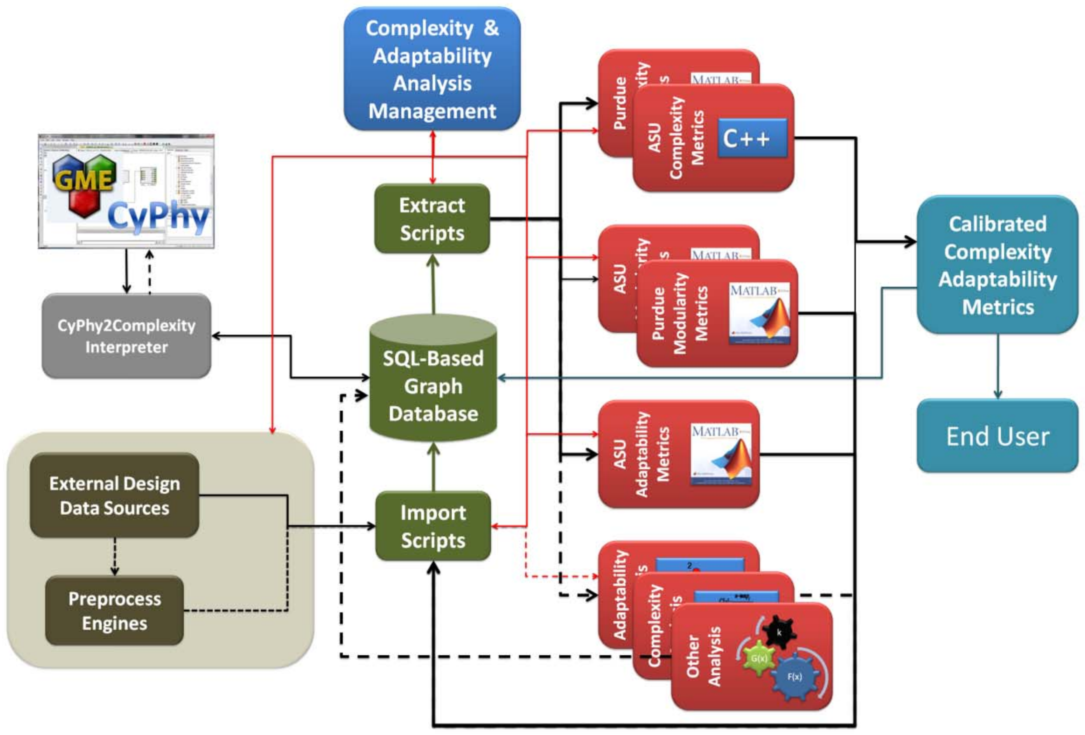


Figure 17 Prototype Metrics Tool Architecture

the graph representations of design artifacts to which the metrics are applied, as well as the measures and metrics that are calculated by the metrics engines. The metrics engines comprise the second major element. These engines compute the various adaptability and complexity measures from graph representations of design artifacts, and the resulting complexity and adaptability functions that combine the measures according to the calibrated estimation functions. The third major element is the interface for extracting graphs from artifacts. The tool supports extracting graphs from a tool defined spreadsheet format (that was used to capture information from legacy design documents during the calibration process), and from models in CyPhyML, the META design language developed by Vanderbilt, via an interpreter for the GME-based CyPhyML editor.

The normalized graph representations provide a common basis from which to extract/build the appropriate metrics analysis engine input data, maintain reference to data source and its identification paradigm, and store the results. The data is stored in a local SQL database to reduce coupling between the components of the tools, and to facilitate transformation of the data from the common directed graph format into the various representations required by the various metrics engines. The graph storage was expressly designed to be simple, high performance and easily hosted on open-source or free database engines. The graph storage also provides persistence for historical analysis, the ability to re-analyze a graph with additional algorithms, and the ability to readily exchange data with other META participants.

The data is stored in a common intermediate format in a local SQL database to facilitate rapid translation of the data into the required representations and formats that the complexity engines require (Figure 18). The graph store provides a library of stored procedures that provide error-checked insertion of the graph data. Stored procedures also provide various data selection

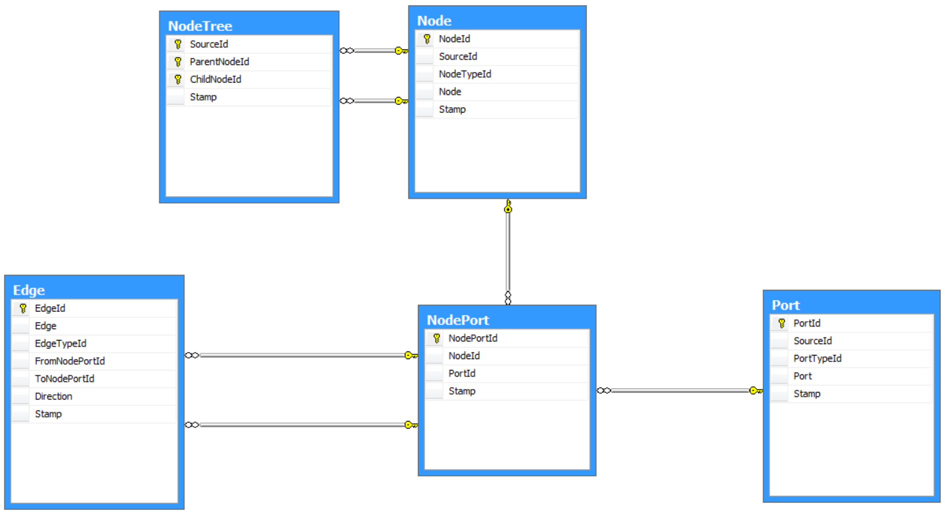


Figure 18 Core Graph Database Schema

functions. The required formatting for the present set of complexity metrics is generated by several stored procedures.

A complete database creation script is provided in the tool distribution. This script build the database, the tables, keys, stored procedures and pre-populates several tables with initial data that provide types and keys such that a simple graph can be constructed immediately (Figure 19).The intermediate representation component provides temporary storage for the constituent system representations (nodes, connections, weightings) in a simplified common model. This common storage is implemented in Microsoft SQL Express, a free SQL engine. The SQL tables and queries have been designed to be portable to other database implementations such as MySQL. SQL provides a flexible and fast method to store, format and extract the graph data for analysis by complexity analysis engines.

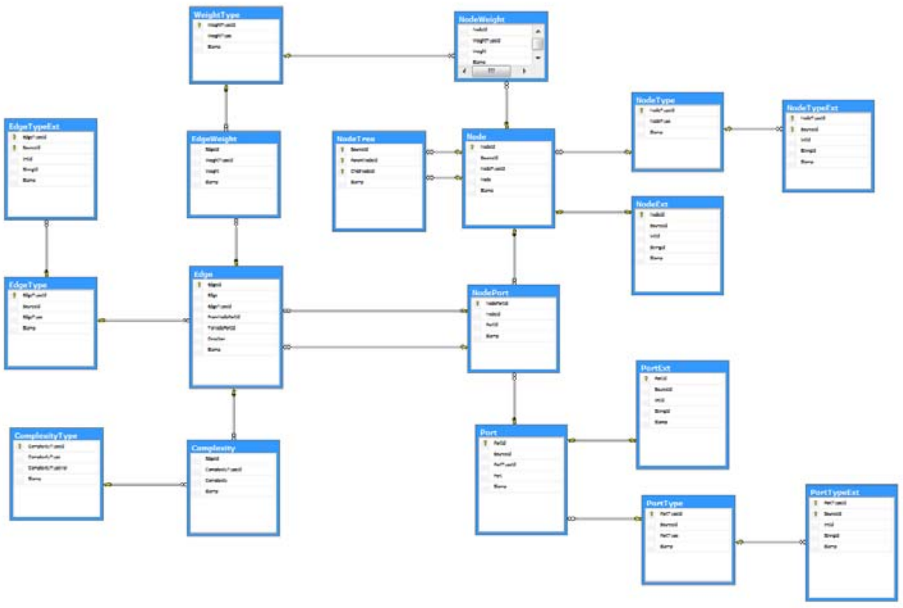


Figure 19 Elaborated Graph Database

The metrics engines represent the second major element of the tool, and provide the primary functionality of the tool. The current set of metrics engines implement the selected complexity and adaptability measures, as well as the functions that combine them into the final complexity and adaptability metrics. Most of the measures in the current implantation have been developed by Arizona State University and Purdue University and are implemented in MATLAB and C++.

The complexity engine manager controls the complexity metrics analysis process. The present prototype includes a flow that is triggered as required by the CyPhy model environment that then extracts the instantiated system model nodes, connections and weights, stores the resulting graph in the database, and uses stored procedures to provide input to and execute the metrics engines and return the resultant metrics back to CyPhy for additional analysis during the design space exploration process.

The tool was designed to import graph representations of differing types of systems, architectures and designs from many different sources (bills of material, enterprise product data management systems, legacy design documents, etc.). The diverse sources of data to which the metrics are to be applied necessitates a flexible import capability supported by readily available, configured, or adaptable applications and tools. The import component extracts or imports system representations, that is, their nodes, interconnections and other attributes.

The present prototype includes two import mechanisms. The first interfaces to CyPhy models in GME via a C++ GME Translator plugin. This plugin (shown as 'Complexity Plugin' in the META CyPhyML GME paradigm) will analyze an instantiated CyPhy model to extract the system nodes, interconnections and weighting of these elements for each domain represented in the CyPhy model. The second import mechanism imports graphs from a tool defined spreadsheet format. This was used during metrics calibration to import data from a variety of legacy data sources.

The tool architecture was chosen to be flexible and extensible. The tool has been implemented as a loosely coupled set of components integrated by the central database to support extensibility. By separating the front end from the internal representation and the metrics engines, new front ends can be developed to support additional modeling languages. Separating the internal representations from the metrics analysis engines enables the easy addition of new metrics as they emerge. It also makes it easy to incorporate the results from individual engines into a calibrated metric, and the retargeting of output of the tool. For example, the tool can be used to provide metrics to a tool for stakeholder visualization or for Multi-Domain Analysis and Optimization (MDAO) by redirecting the output.

New input formats can be supported by simply developing a SQL exporter for the format. Such exporters could be implemented directly via SQL scripts, programmatically via database interfaces (e.g., Java Database Connectivity (JDBC) for Java programs, ODBC for C++), or via export to intermediate formats such as text files.

Metrics components can likewise be developed and integrated using a variety of techniques. The current prototype includes metrics engine components implemented in MATLAB and C++. The distribution also includes examples of integrating metrics engines programmatically via C# and Component Object Model (COM) interfaces, via text files and shell scripts, and via SQL scripts.

The tool distribution includes the source code for all of the metrics engine components and the CyPhyML interpreter, and various helper applications, as well as the SQL scripts needed to manage the database, and various scripts and stored procedures that provide the interface between the front ends and the database, and the database and the metrics engines. The distribution also includes Readme files for all elements of the tool, as well as a User Guide and a Configuration Guide. The configuration guide will provide information to support the extensions described here (new front ends, new metrics, new output targets).

The tool makes use of a number of commercial off the shelf (COTS) and/or open source tools at both build and execution time. These include the Microsoft SQL Express 2008 r2 free SQL server, SQL client tools such as the Microsoft SQL Server Management Studio and Microsoft Data Access Components (MDAC) or the TOAD freeware tools, Microsoft Visual Studio, Mathworks MATLAB, and Microsoft Excel 2007. All of these dependencies are documented in the tool documentation. Open source substitutes for the SQL server and client tools, Excel, and MATLAB could be substituted if desired, though a degree of adaptation may be required.

4.5. Experimentation

The experiments we have performed in the program fall into two broad categories. First are metrics based experiments that sought to exercise and evaluate various individual metrics. The second category is integration experiments that demonstrate and validate our approach to using the metric. The remainder of this section discusses experiments conducted during the program, organized according to these categories. The first subsection covers metrics based

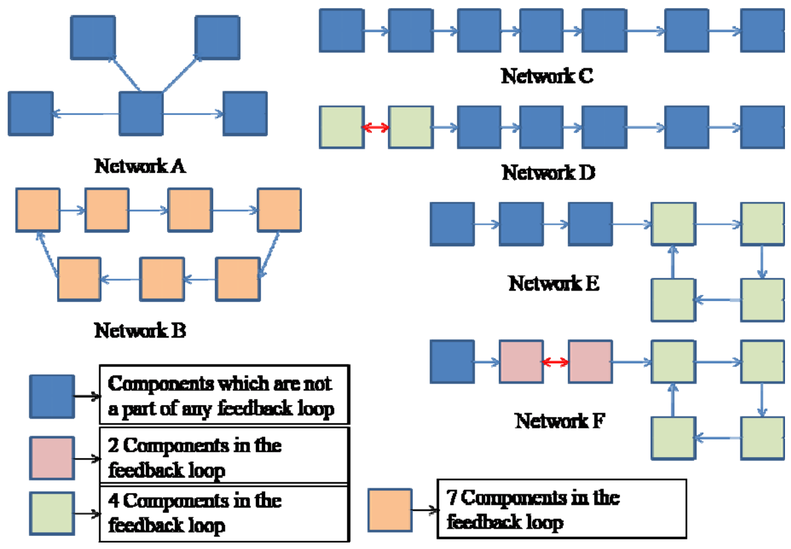

Network C
Network A
Network D
Network E
Network B
Network F
Components which are not a part of any feedback loop
2 Components in the feedback loop
4 Components in the feedback loop
7 Components in the feedback loop

Figure 20 Synthetic Examples

experiments and the second covers integration experiments.

4.5.1. Metrics Based Experiments

The first metrics based experiments we describe focus on applying metrics to synthetic examples. Figure 20 shows synthetic examples which were created to demonstrate the effectiveness of the size and coupling metrics. For the ease of demonstration, nodes and links are assumed to be of unit complexity.

Table 10 describes the complexity scores for each of the networks shown in Figure 20. Increasing the number of nodes will result in increase in size and total complexity. Another important observation is that as the networks with feedback have significantly higher coupling than those without feedback. In addition, complexity significantly increases with increase in number of components within a feedback cycle. Thus, network B has significantly higher coupling than networks E or F. This is expected, as increasing the number of components in a feedback path makes it increasingly difficult to accurately predict the behavior of the system.

While the synthetic examples demonstrate the reasonableness of the metrics and the proposed scheme for the aggregation of different aspects of complexity, a real aerospace design example is required to demonstrate some of the merits and demerits of the proposed framework.

Table 10 Calculating Complexity of Synthetic Examples

Networks	Size Complexity	Coupling Complexity	Overall Complexity
A	8.04	4	12.04
B	13.62	1029	1042.62
C	13.62	56	69.62
D	13.62	64	77.62
E	13.62	150	163.52
F	13.62	208	221.62

For this purpose three existing satellite missions were chosen for analysis. The choice of these missions was primarily driven by availability of the relevant information about the structural graphs. Structural graph for these missions can be found in "Reducing Space Mission Cost" by Wertz and Larson [26]. For this analysis it is assumed that the components correspond to the nodes of the structural graph and the interactions between the components correspond to the links. Due to the lack of availability of the functional data the weights of the all the nodes and links are assumed to be one. Four different types of interactions have been considered while creating the structural graph namely, matter, energy, force and information. For this analysis all these interactions have been assigned equal importance.

Figure 21shows the structural graph for Orsted satellite mission.

As described in the previous section, any complexity analysis begins with choosing an appropriate level of abstraction. For this problem, we have chosen to analyze the system in the component level of abstraction. The choice of component level of abstraction is dictated by the fact that all the chosen satellites look almost identical at subsystem level of abstraction.

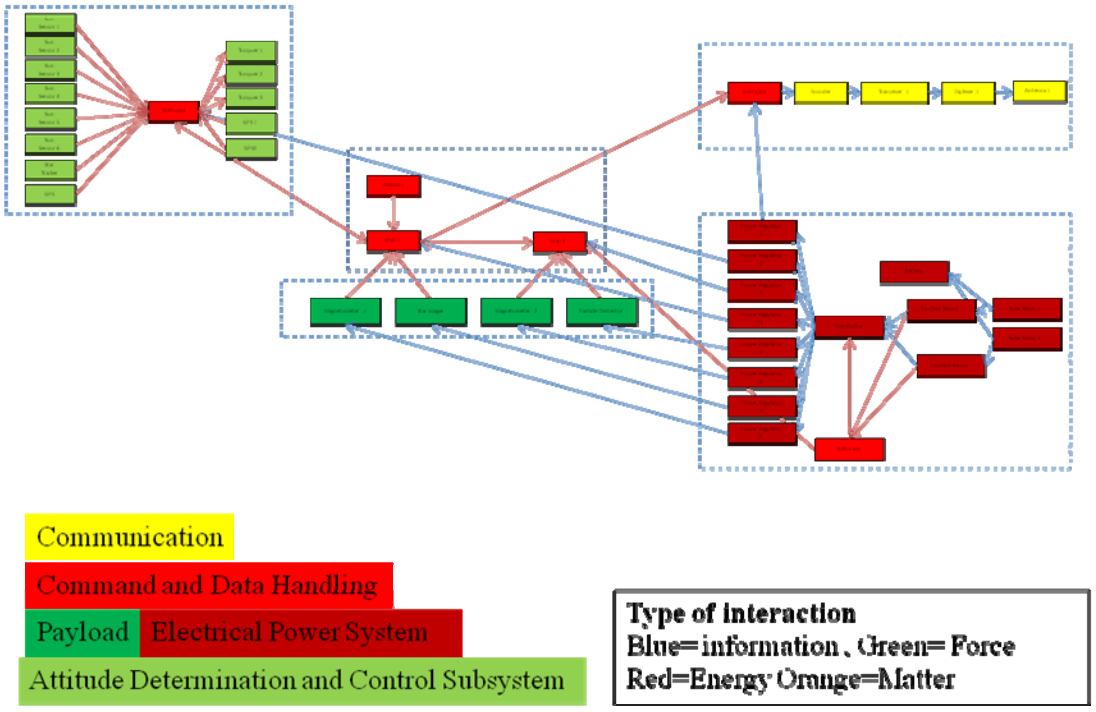

Communication
Command and Data Handling
Payload
Electrical Power System
Attitude Determination and Control Subsystem
Type of interaction
Blue=information , Green= Force
Red=Energy Orange=Matter

Figure 21 Structural graph for Orsted Satellite

Table 11 shows the relevant mission about the example satellites. It also shows the size and coupling complexity of the three missions. It is found that for the chosen example set the metrics for complexity show a promising correlation with the development cost of the mission (Figure 22). Figure 23 shows the complexity of the individual subsystems as well as the integration complexity. It can be observed that integration complexity is higher than complexity of all the subsystems. This is expected for any complex aerospace system where complexity of integration is major driver to cost and schedule. It is also seen that complexity of Clementine is significantly higher. This can be attributed to the fact that power requirements for Clementine are a major design driver and hence resulted in a complex power subsystem. This demonstrates an important fact that performance and complexity are correlated and components intentionally coupled to extract high performance from the system.

Table 11 Complexity of the example satellites

	Orsted	HETE	Clementine
Mission	Earth Magnetic Field Measurement	Multiwavelength study of gamma ray burst	Observation of moon and asteroid 1620 Geographos
(Total Cost – Launch cost) (Million dollar)	15	23	60
Weight (Kg)	60	125	232
Size(cm)	45 X 34 X 68	90 X 50 X 50	Diameter:110, Height:120
Size Complexity	499.9	621.1	818
Coupling Complexity	4893	7749	14962

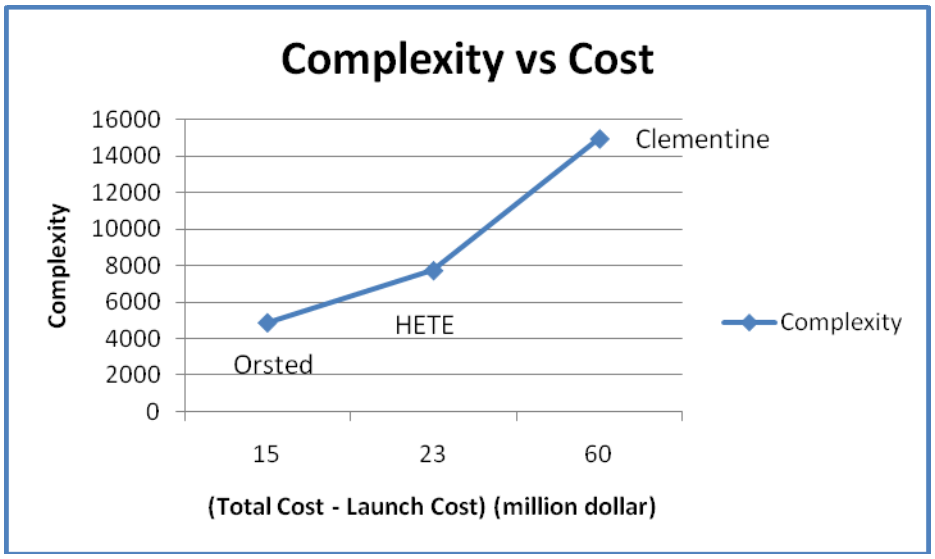


Figure 22 Correlation of Cost with Complexity

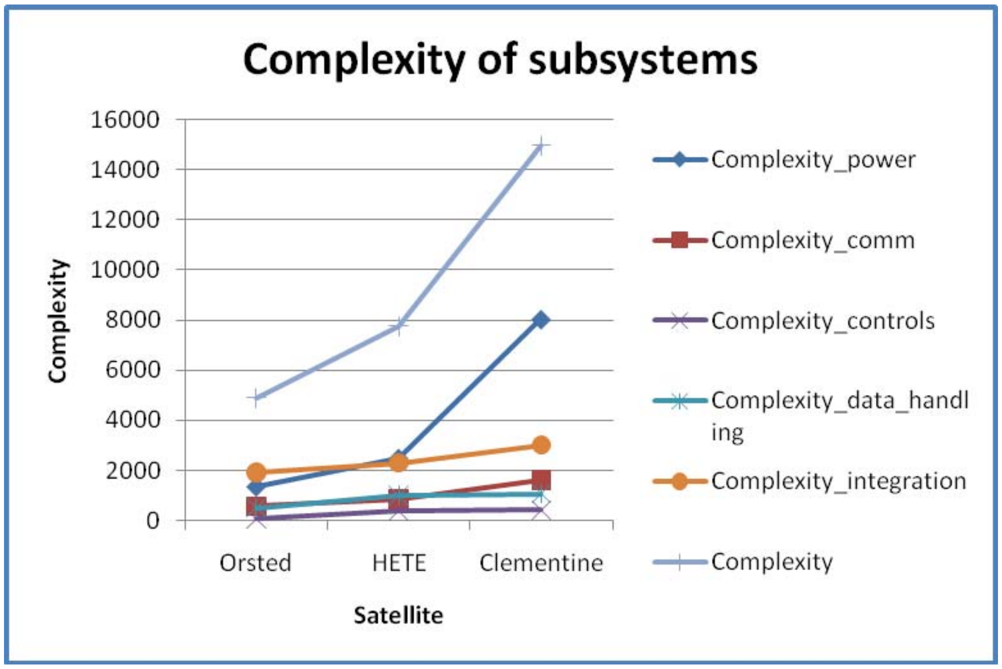


Figure 23 Correlation of Subsystems and Integration

The examples described in the previous sections are relatively small. Most modern aerospace systems contain thousands of components and millions of interactions. Hence it is important to demonstrate the scalability of the proposed metric for these large scale systems. Since the size measure only involves counting the number of components and interactions it is easily scalable for large networks. The proposed measure for coupling requires calculation of all the paths between the two nodes and hence the computation time increases exponentially as we increase the number of nodes and links in the system. Hence an approximation is made to ensure the scalability of the coupling metric. It can be seen that step 2 of the proposed algorithm, which involves calculating all the dependencies (all the paths) between all the node pairs in the network, is the most computationally intensive task. Hence in this step instead of examining dependencies between all the node pairs of the network we assume that the influence of a particular node is restricted only to those nodes which are reachable within 50 'hops' of that particular node. Figure 24 shows an example network where the radius of influence is assumed to be 2 'hops'. Hence all nodes which are reachable within 2 'hops' from node I are examined, shown in green, influence on other nodes, shown in red, is ignored.

To demonstrate the scalability of the metric with this approximation several synthetic examples were constructed. Random graphs of different sizes were created and the time required by calculating the coupling metric is examined. The results are summarized in Table 12.

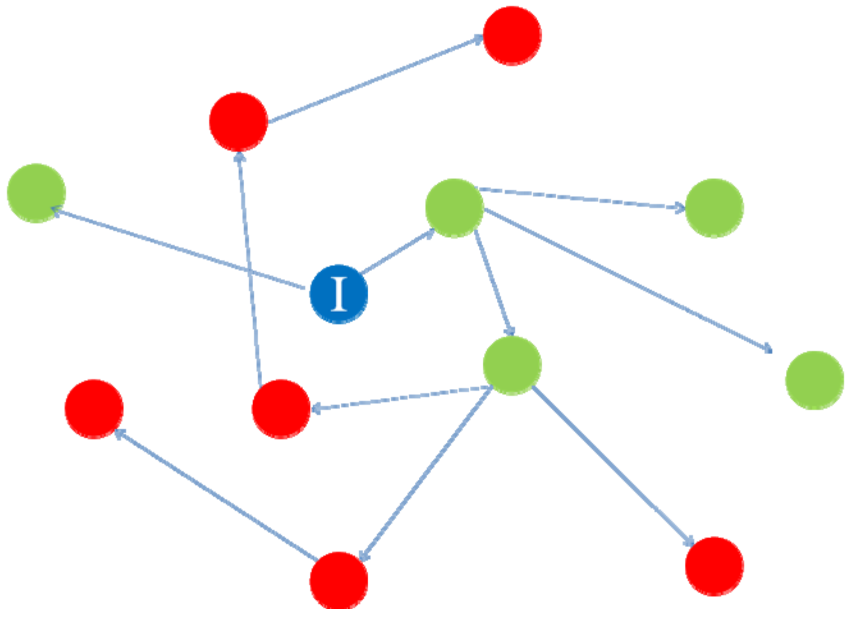
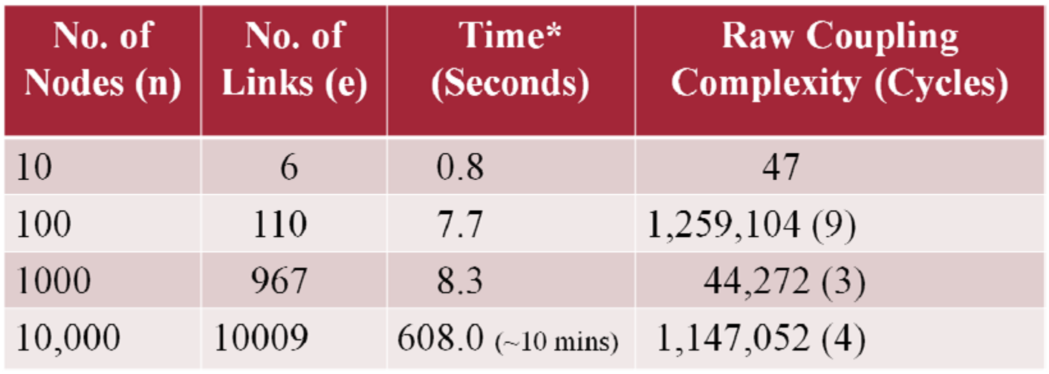


Figure 24 Nodes Within the Radius of Influence (2 Hops) of Node I

Table 12 Demonstrating the scalability of the coupling metric



No. of Nodes (n)	No. of Links (e)	Time* (Seconds)	Raw Coupling Complexity (Cycles)
10	6	0.8	47
100	110	7.7	1,259,104 (9)
1000	967	8.3	44,272 (3)
10,000	10009	608.0 (~10 mins)	1,147,052 (4)

Random networks are constructed with number of nodes (n) and link probability (p) such that np=1. This is a reasonable assumption as these satellite examples had np≈1.2. Table 12 shows results from a moderately optimized code executed in MATLAB. All the examples were run on 8 core (0.8 GHz each) 16Gb RAM, 2x Quad-Core AMD Opteron 2380 system. It is found that with the above approximation the Overall, computational time is reduced to $O(50n(np^{50})$ from $O(n^2(np)^n)$. Hence the metric is scalable (O(n)) as long as value of np, which represents the average degree of the network remains close to one. Future work will focus on demonstrating the scalability on a large existing aerospace system.

Speed reduction ratio (20:1)
I/O shaft relative orientation (90 deg)
I/O rotation direction: either
Power: Motor power 5 hp, 2000 rpm, I/O location (as shown)
Design criteria: Cost, heat loss, total volume

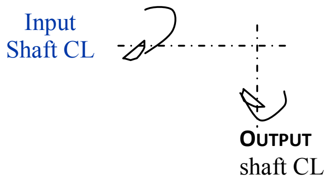
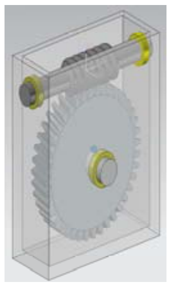
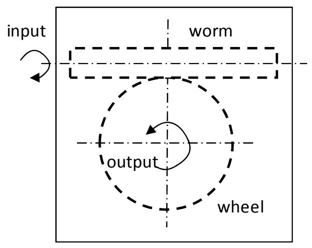


Figure 25 Requirements

The next experiment involved applying a subset of the candidate complexity metrics to a set of alternative designs for a purely mechanical system. A synthetic problem was defined



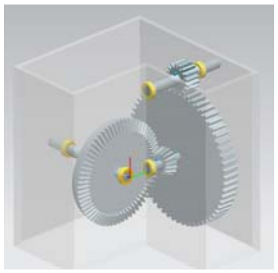


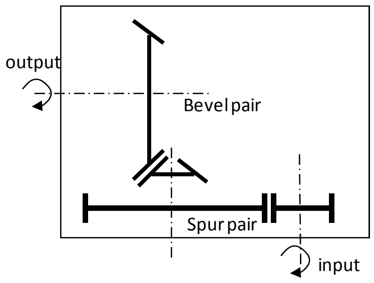
Advantages
•allows high gear ratio with shafts intersecting at 90 degree; need only one gear pair
•The size of the worm wheel gear box is smallest for high gear ratio.
•Low noise during running because of constant meshing between worm and wheel.

Drawbacks
•Low efficiency; maximum heat loss due to sliding between the faces of gears.
•Bearings needed for both radial and axial loads.

Figure 26 Design 1 - Worm Gear

(mechanical transmissions) and requirements specified. Four alternative designs, all capable of meeting the specified requirements, were generated. All design parameters were identified and their values for each alternative were calculated. From detailed designs and plans for each alternative, artifact functional structure, design process structure and manufacturing operations and sequences were determined and represented in the form of entity relation graphs. These graphs were then used to compute various measures of complexity. The requirements are shown in Figure 25.





Advantages

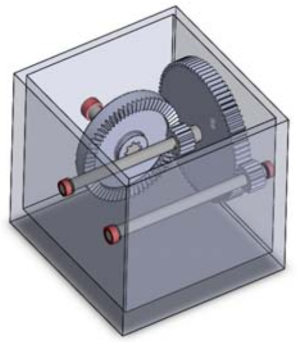
- high efficiency (straight teeth)
- easy to machine and assemble.
- For two stages, Spur gears can carry higher speed ratio than the bevel gears to allow compact design.
- Heat losses are low compared to worm gears

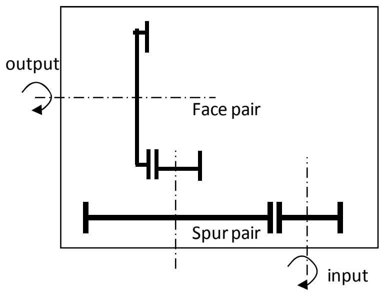
Drawbacks

- Spur gears are noisy at very high speeds and straight bevel gears are also noisy beyond a certain speed.
- Occupies greater volume than Alt 1
- Both radial and thrust bearings are required

Figure 27 Design 2 – Bevel Pair

Four alternative designs were generated, all to meet the same reduction ratio and input Hp. All components in each design were sized and analyzed for performance and structural failure. The conceptual layouts are shown below and their advantages/disadvantages are listed in





Advantages

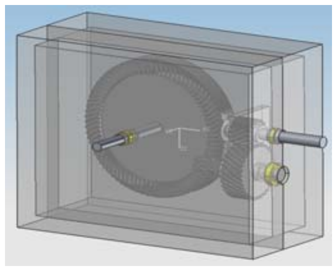
- all the gears are either Spur or equivalent of a spur gear it is easy to Design and manufacture compared to the Bevel-Spur gear or the Hypoid-Helical gear setup.
- Heat loss is low compared to worm gears
- No thrust bearings are required.

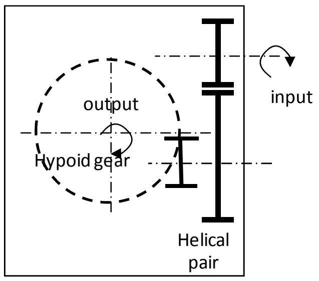
Drawbacks
- very noisy compared to all other gearboxes
- gearbox is much larger than Alts 1 and 2.

Figure 28 Design 3 – Face Pair

Figure 26 for design 1, Figure 27 for design 2, Figure 28 for design 3 and Figure 29 for design.4

Standard gear design formulas were used. Structural and kinematic analysis was done to determine the dimensions of all components (gears, shafts, keys), gear teeth geometry and part locations. Bearings types were determined but COTS were not selected. Complexity metrics were calculated for the architecture for each design, as well as for the design and manufacturing processes associated the designs. Results are presented in Table 13 for functional complexity, Table 14 for design process complexity, and Table 15 for manufacturing complexity.





Advantages

- Hypoid gear is used where high reduction ratio (up to 15:1) and torque is required.
- Both hypoid and helical gear train are less noisy compared to spur and bevel gear train.
- efficiency is less than the bevel and spur gear but more than the worm wheel gear.

Disadvantages

- Both hypoid and helical gear train are difficult to manufacture (complicated tooth profiles).
- Both hypoid and helical require radial and thrust bearings

Figure 29 Design 4 – Hypoid Gear

Table 14 Functional Complexity of Gear Designs

Metrics	Design1	Design2	Design3	Design4
# nodes	37	77	85	81
# links	118	212	185	246
Bridge Nodes	1	6	9	6
Average node degree (a_{AVG})	6.29	5.45	4.34	6.07
Maximum node degree (a_{MAX})	12	16	17	20
Connectedness	0.17	0.07	0.05	0.07
I_{VD}	654.1	1130.9	883.2	1428.6
N I_{VD}	0.09	0.03	0.02	0.03
Clustering coefficient (C_{AVG})	0.33	0.39	0.47	0.41
Mobility	277	479	382	573

Table 13 Design Process Complexity of the Gear Designs

Metric	Design1	Design2	Design3	Design4
# nodes	22	34	33	35
# links	31	53	51	55
# Bridge Nodes	3	3	3	4
Average node degree (a_{AVG})	2.81	3.11	3.09	3.14
Maximum node degree (a_{MAX})	7	7	7	7
Connectedness	0.13	0.09	0.09	0.09
I_{VD}	103.79	190.53	182.53	199.75
N I_{VD}	0.05	0.03	0.03	0.03
Clustering coefficient (C_{AVG})	0.60	0.55	0.55	0.54
Total Process Time (min)	99	195	192	196
Mobility	46	88	84	92

Table 15 Manufacturing Process Complexity of Gear Designs

Metric	Design1	Design2	Design3	Design4
# nodes	24	56	52	56
# links	23	55	51	55
Average node degree (a_{AVG})	1.91	1.96	1.96	1.96
Maximum node degree (a_{MAX})	4	4	3	4
Connectedness	0.08	0.03	0.04	0.03
I_{VD}	47	119	106	119
N I_{VD}	0.02	0.006	0.007	0.006
Clustering coefficient (C_{AVG})	0.75	0.72	0.70	0.72
Machining time (min)	37.62	162	155.18	265.44
Machining Cost ($)	489.07	1845	1,789.25	2,760.75
Mobility	18	50	46	50

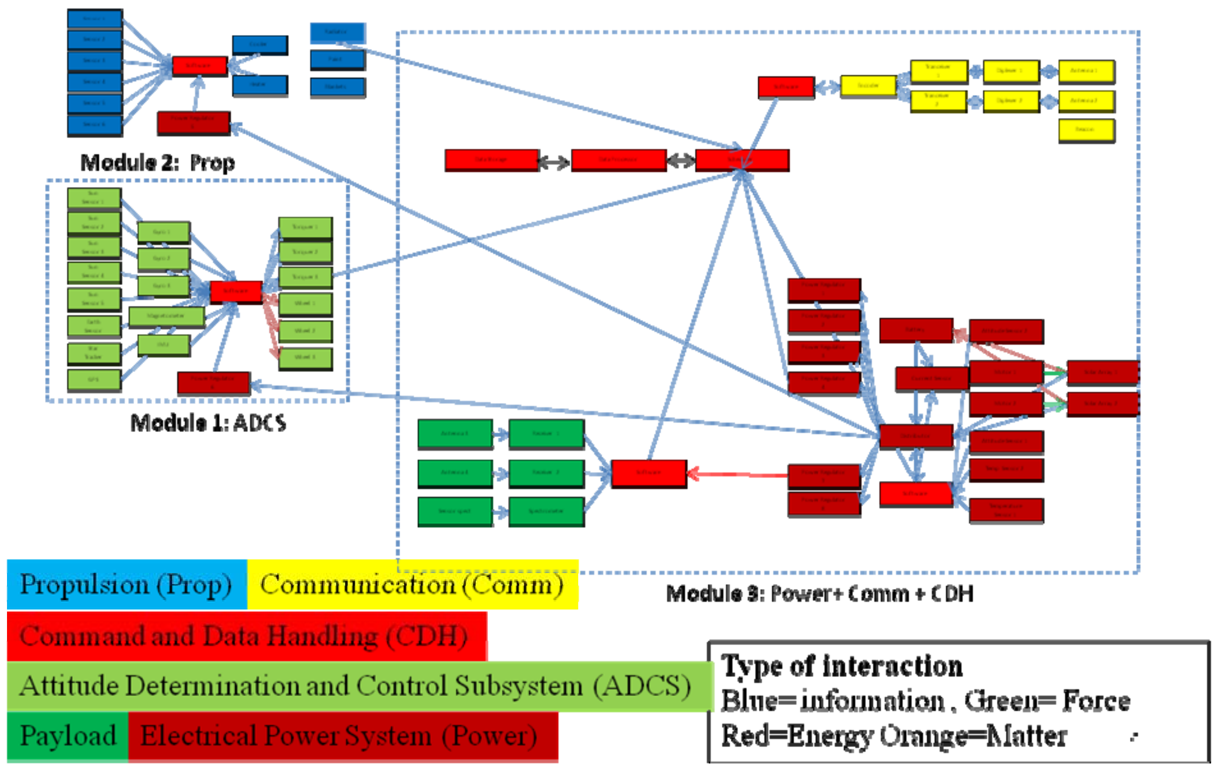

Module 2: Prop
Module 1: ADCS
Module 3: Power+ Comm + CDH
Propulsion (Prop)
Communication (Comm)
Command and Data Handling (CDH)
Attitude Determination and Control Subsystem (ADCS)
Payload
Electrical Power System (Power)
Type of interaction
Blue= information , Green= Force
Red=Energy Orange=Matter

Figure 30 Optimal Modules for HETE Satellite

The next experiment applied the new Purdue modularity metric to the satellite examples discussed previously to illustrate the modified approach for optimal modules and exercise of the resulting modularity metric. The optimal decomposition obtained for the HETE spacecraft (Figure 30) and the others (Table 16) make clear some observations. First, the result suggests that most of the components of propulsion and control subsystems of Clementine should be in a single module. This is justified since Clementine's mission is to perform scientific observations of moon and near earth asteroids; thus, control input is continuously required for orbital maneuvers. Second, the results suggest that the power regulator should be a part of each subsystem (Controls, Payload, Communication, propulsion) instead of part of the power subsystem. Overall, the complexity is reduced in all three cases when the optimal modules are

Table 16 Modularity Results

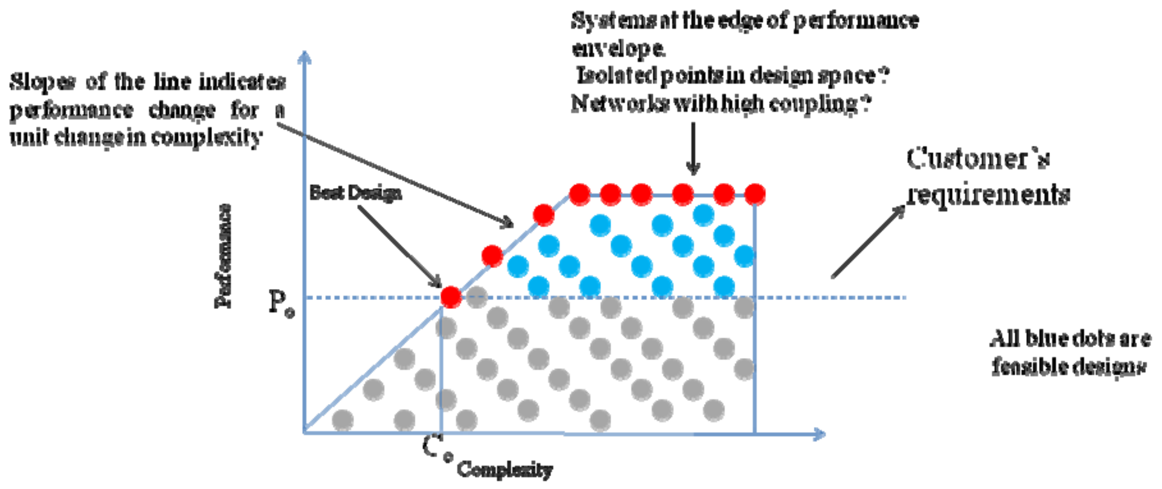
Clementine		**HETE**		**Pluto**	
Modules and number of components	**Complexity**	**Modules and number of components**	**Complexity**	**Modules and number of components**	**Complexity**
Power + CDH + Pay load (35)	60413	Power + CDH + Payload + comm. (28)	5432	Power + CDH + Payload (24)	5896
Power regulator + Comm (6)	1225	Power regulator + prop(9)	808	Power regulator + Prop (11)	1135
Prop + ADCS (28)	1710	Power regulator + ADCS (22)	510	Power regulator + ADCS (25)	732
				Power regulator + Comm. (10)	1843
Integration	1556	Integration	999	Integration	2223
Complexity Without Module	64904	**Complexity Without Module**	7749	**Complexity Without Module**	11829
Complexity with modules	63385.3	**Complexity with modules**	6878.7	**Complexity with modules**	10023.7

determined and used to estimate system complexity.

4.5.2. Integration Based Experiments

This first integration based experiment uses a notional example to show how metrics for complexity and adaptability could be used to assist in design space exploration. Figure 31 describes a notional curve between performance and complexity for a given design problem with a finite component library. As seen in the figure, for a given complexity, C_0, the performance of a design created from a given component library is limited by P_0. The slope of the line indicates a trade-off between performance and complexity. The best design is the one which meets customer requirements with minimum complexity. The aim of complexity enabled design space exploration is the identification of designs which minimize complexity while providing satisfactory performance.

To demonstrate the validity of the notional figure shown in Figure 31 we have chosen a circuit design problem. A circuit problem has been chosen because the performance of electrical circuits can be easily quantified. The primary goal of the problem is to design a low-pass filter circuit with cut-off frequency between 5 and 15 Hz. The designs which result in frequencies close to 10 Hz are preferred. Low-pass filters are those circuits which exhibit a unity gain at all frequencies below the cutoff frequency and attenuate the frequencies which are greater than the cutoff frequency. There are two steps in generating each design. The first one involves choosing a set of components from the component library. The component library is a collection of components which can be used to create different circuit designs. Table 17 shows the component library used in this problem. The zero value of the resistor in Table 17 indicates a short circuit



Exploring the entire design space for a variety synthetic problems will enable us to identify network structures which are invariant across different problems.

Figure 31 Performance vs. Complexity for a design with finite component library

while infinity indicates an open circuit. These two components are neglected while calculating complexity as they do not contribute to system complexity.

Once the components have been chosen their connectivity is represented using a template circuit shown in Figure 32. Each black box in the template can be replaced by a component chosen from the component library (Table 17).

Table 17 Component library for the circuit problem

Type\Value				
Resistor (R)	0 Ω	1 Ω	100 Ω	∞ Ω
Inductor (L)	1μH	1mH	1H	
Capacitor (C)	1μF	1mF	1F	

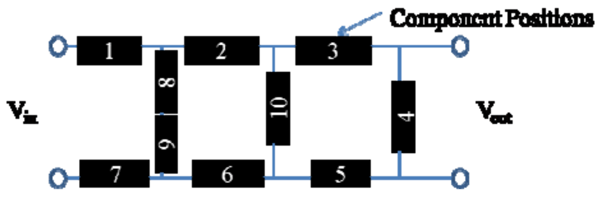


Figure 32 Template circuit for the example problem

There is no limitation on the number of components of a particular type. Once the circuit design is completed its cutoff frequency is computed using WinSpice [9]. WinSpice is a commercially available circuit solver which can simulate a wide variety of electrical circuits. The performance of the circuit is defined by (30).

$$P = \begin{Bmatrix} v - 5 & for\ 5 \le v \le 10 \\ 15 - v & for\ 10 < v \le 15 \\ -1 & otherwise \end{Bmatrix} \quad (30)$$

Here P is the performance of the circuit while v is the cutoff frequency which is defined by (31).

$$v = \begin{Bmatrix} v_{lp}\ for\ Low\ Pass\ filters \\ 0\ otherwise \end{Bmatrix} \quad (31)$$

Here v_{lp} is the cutoff frequency for the designs which correspond to low pass filters. The choice of the performance function is motivated by the fact that it results in positive performance between 5 and 15 Hz with performance being maximum at 10 Hz. All the designs which are not within the design range or are not low pass filters are penalized.

Since design space for the problem is extremely large (~10^{11} designs) an exhaustive design space exploration is not feasible. Further, the design space is multi-modal and discrete local gradient based explorations may not be best suited for this analysis. Hence a genetic algorithm based exploration scheme is adopted. In order to develop a performance vs. complexity graph similar to Figure 31 exploration of different regions of the design space is required. In order to guide the algorithm to discover designs laying in a particular region of the design space the fitness function is modified such that designs within the region have high fitness. For instance if all designs with performance greater than 4 are desired, the objective function is modified to provide higher fitness to these designs. The process starts with a random population of size 100 which than evolves over 100 generations and results in designs which have higher fitness. This process generates several distinct designs, which lie in the region of interest. The process is repeated several times over different regions to generate performance-complexity chart shown in Figure 34. Several designs within the design space were found to have identical performance and complexity. In order to distinguish between these designs, a small random number is added to the performance and complexity of each design. A scaling factor of 10 is applied to (1) to help in differentiating between the complexities of the designs. The solid triangles in the figure show the performance-complexity plot when random number is added. The hollow triangles on the other hand show the same relationship without the random number.

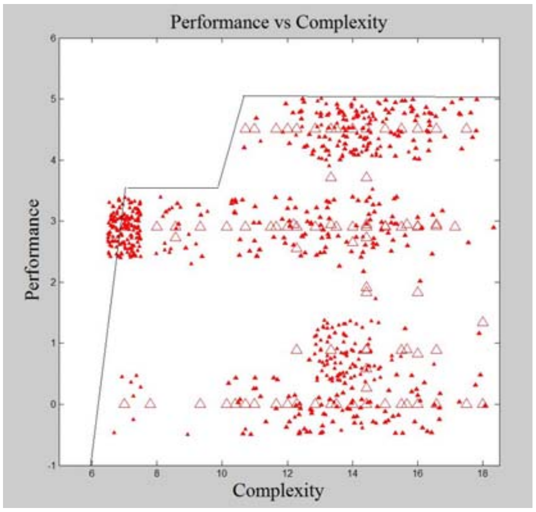


Figure 34 Performance vs. Complexity Curve Generated Through Complexity Enabled Design Space Exploration

It is evident from the graph that the exploration is successful in generating performance-complexity curve similar to Figure 31. The difference between the two figures is that variation between performance and complexity occurs in discrete levels Figure 34 as opposed to the smooth slope observed in Figure 31. This can be attributed to the discrete component library and finite size of the template circuit. The primary area of interest in the design space is the high performance region. It can be seen that there exists a range of designs which vary significantly in their complexity but provide similar performance. In order to compare the proposed approach with performance-based design space exploration, we conducted the same exploration again, but

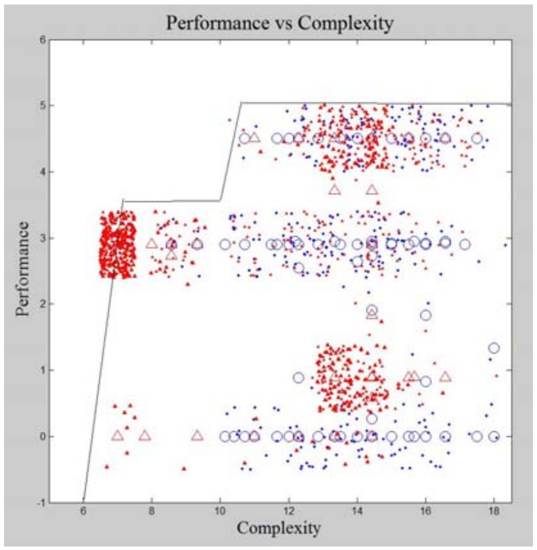


Figure 33 Performance vs. Complexity Curve: Red are Complexity and Performance in Objective Function; Blue are Performance Only

without an explicit requirement on complexity in the objective function.

A comparison between the results obtained from the two design explorations is shown in Figure 33. Results after taking out complexity from the fitness function are shown in blue circles and the red triangles are extracted from results generated with complexity in the fitness function (Figure 34). Figure 33 shows that several low complexity regions are easier to obtain with complexity enabled design space exploration. With a quantitative metric for complexity the exploration algorithm can be specifically guided towards regions of interest and identify the designs which have a desirable mix of performance and complexity. An explicit choice of low complexity designs can lead to design of systems which are not only cheaper and faster to build but provide better predictability in development cost, schedule. In addition, an analysis of designs shown in Figure 34 and Figure 33 can also lead to identification of features which result in good or bad designs. These lessons learnt can help in future design explorations.

Figure 35 and Figure 36 describe two contrasting designs which were obtained from the design space exploration. Since size was the primary aspect driving system complexity, designs

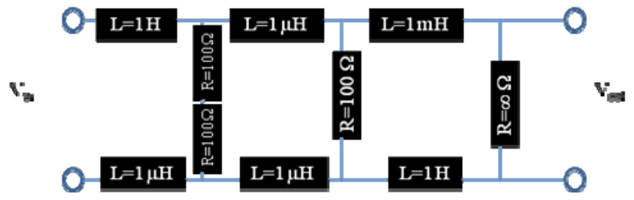


Figure 35 Design with High Performance High Complexity (Performance=4.51, Complexity=17.05)

which have more number of components have proved to be the one with higher complexity. Also both designs have been found to have similar performance.

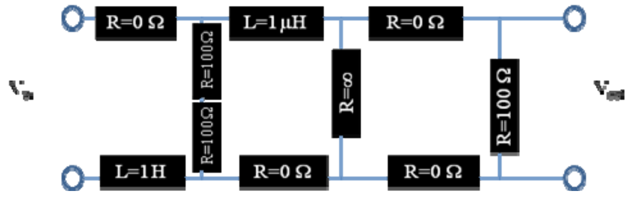


Figure 36 Design with high performance and low complexity (P=4.37, C=10.93)

For notional RLC circuit example described in the previous section we also explore some aspects of adaptability in designs. One of these explorations involves investigating what happens if during the design process, the configuration of one of the components was changed, and the system still remains a low pass filter, i.e. one of the resistors became a capacitor/inductor or a resistor was swapped with another resistor of different resistance. This falls under the category of Type-II Robustness described in the previous sections. Another aspect which we wish to investigate is called Type III Modularity, i.e. changing one component allows the system to change its characteristics from a low pass filter to some other type of filter.

These insights are obtained by exploring the neighborhood of the designs. For the design problem under consideration, each design has 90 designs which lie in its nearest neighborhood. This means that each of these 90 designs can be obtained by modifying one of the design variables while keeping all others constant Figure 37 describes the proposed approach for exploring the neighborhood of a particular design. The blue circles represents the baseline design, green circles indicate the designs which are low pass filters, pink circles show the designs which lie outside the required frequency range and red designs indicates the designs which do not correspond to any type of filter. The size of the green circles is indicative of the difference in performance between the baseline design and the green design. A robust design is one in which the fraction of green design is significantly higher than the designs of other colors similarly a highly modular design is one where the fraction of pink designs is high. Figure 37 and Figure 38 show the cases of high robustness and modularity respectively. These designs have been obtained by searching the neighborhood of the 170 designs which exhibited high performance. Future work will be focused on a thorough exploration of the design space to characterize the relationship between flexibility and adaptability.

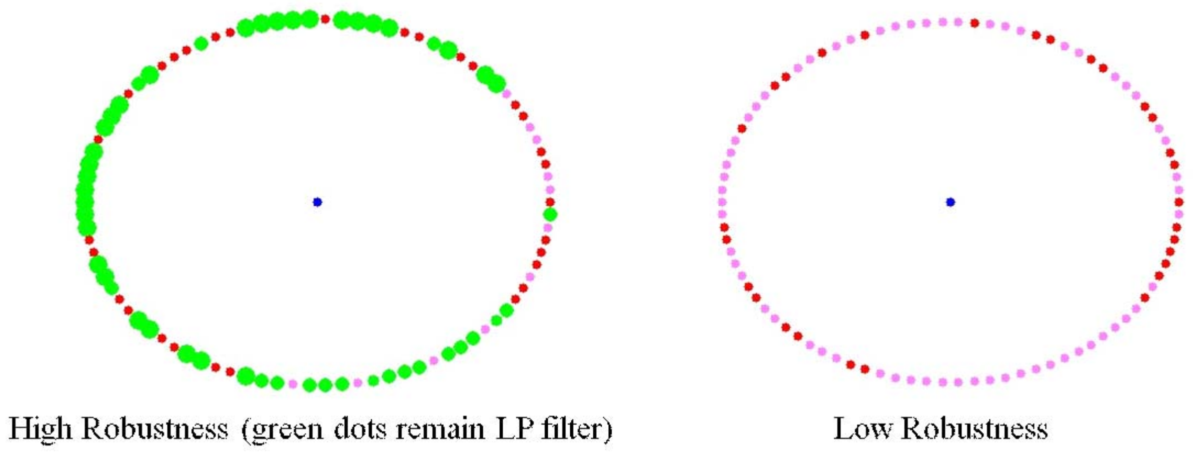


Figure 37 Exploring robustness adaptability of a design

Figure 39 and Figure 40 show the preliminary results of the design space exploration with robustness and modularity as the design objective along with performance and complexity.

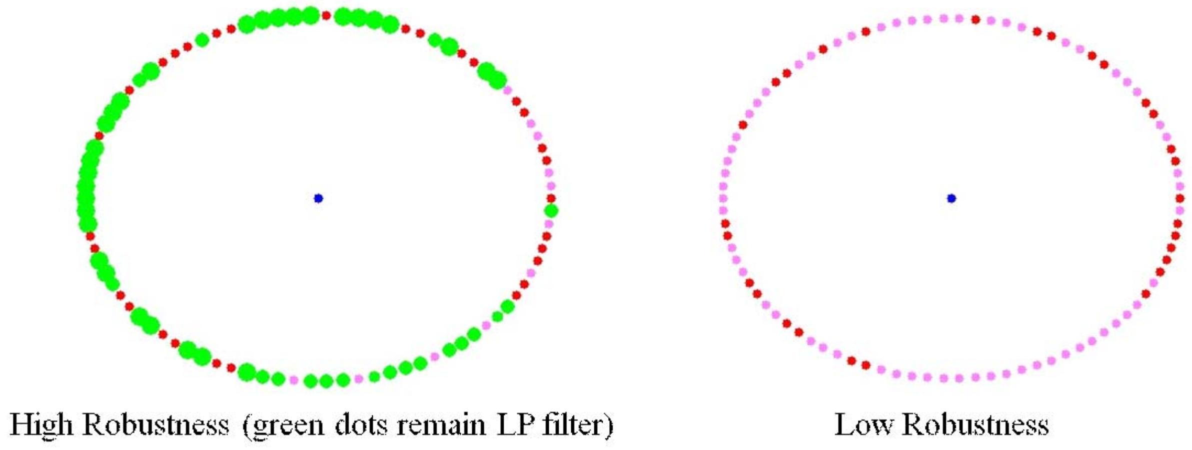


Figure 38 Exploring Modularity Adaptability

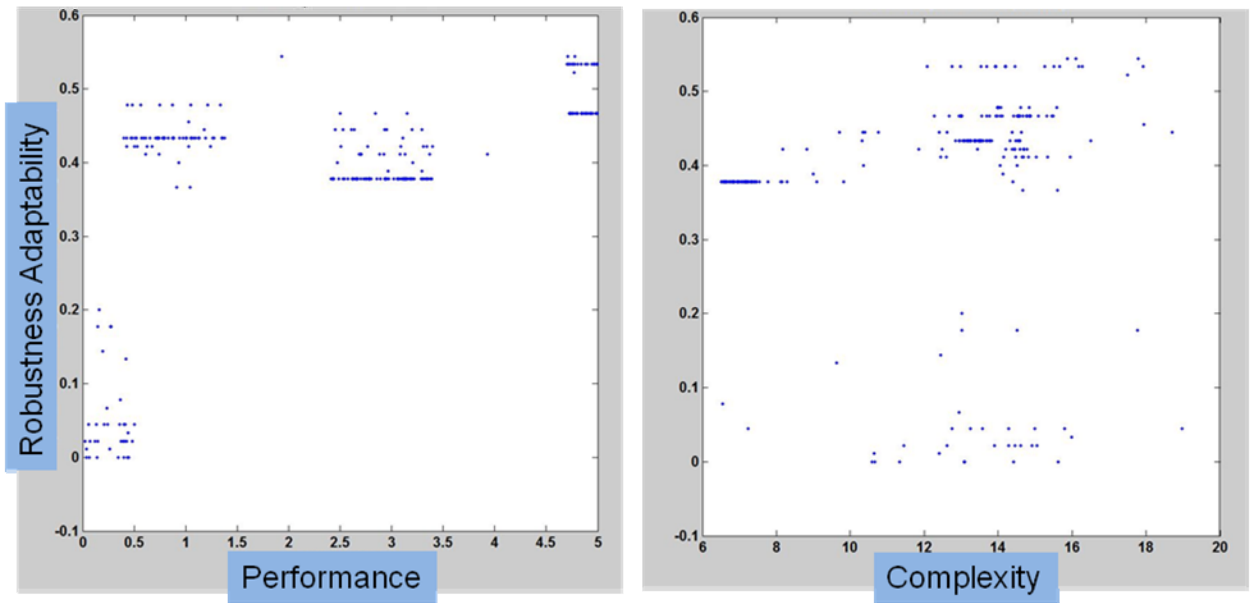


Figure 39 Robustness-adaptability vs Performance and Complexity

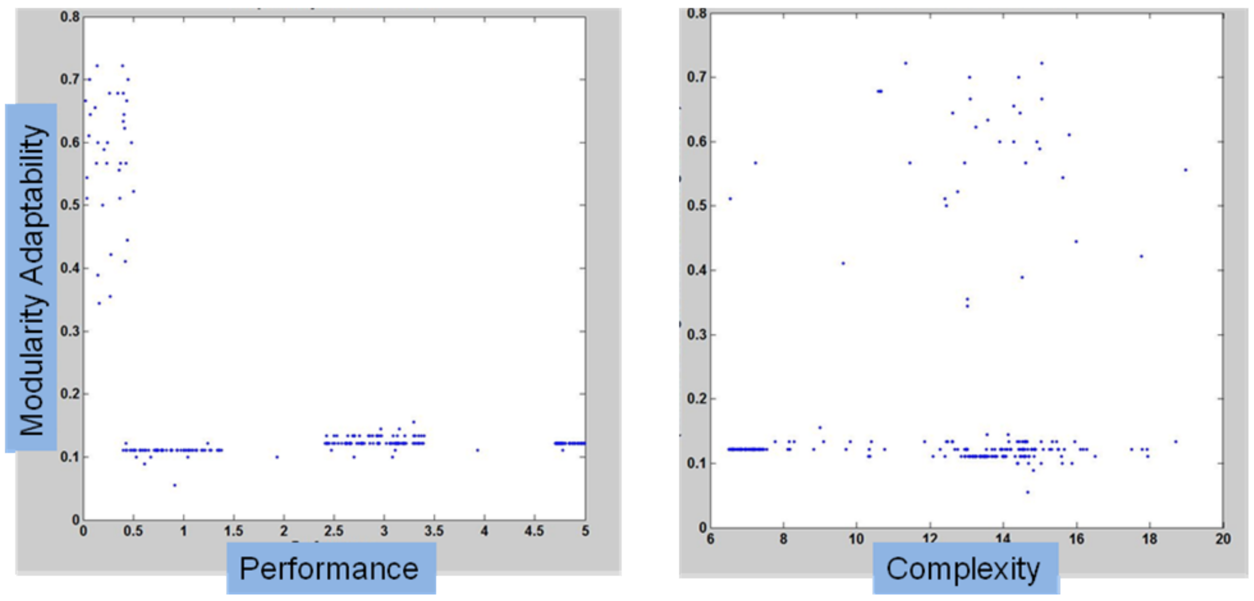


Figure 40 Adaptability vs Performance and Complexity

This next integration based experiments considers how metrics can be used as integral part of a MDAO phase of design space exploration. For this experiment we developed a simple small unmanned aerial vehicle (SUAV) challenge (Figure 41).

The SUAV challenge problem focused on the design of a small tube launched UAV with

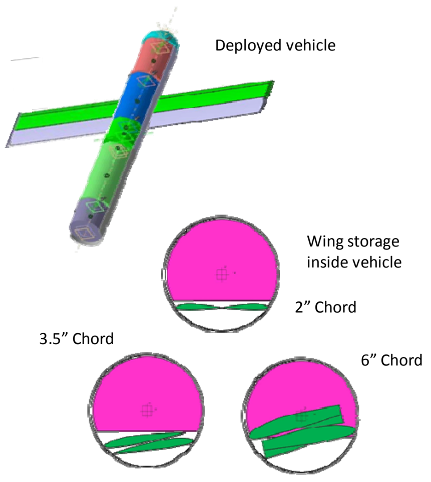
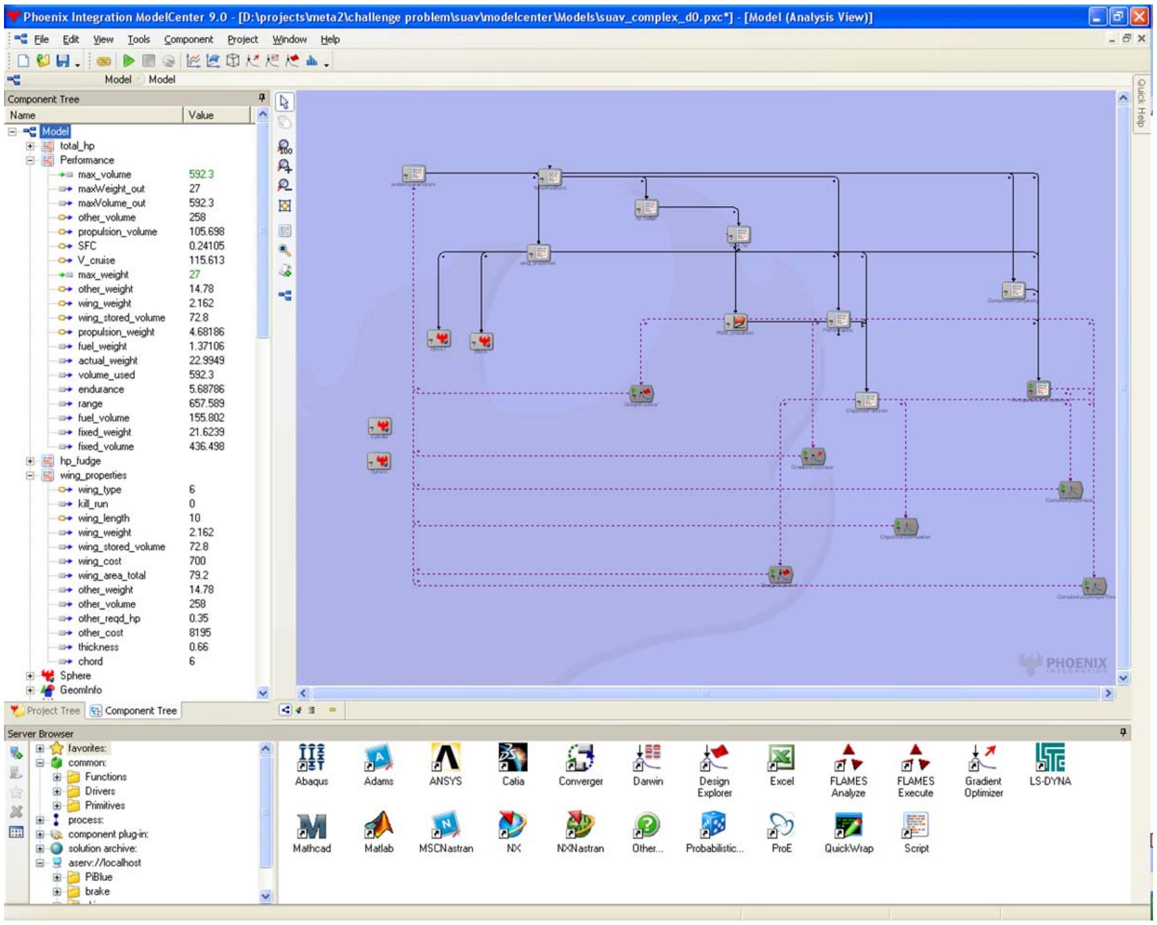


Figure 41 SUAV Challenge Problem

range and endurance requirements, and volumetric and weight constraints. This presented three principal design trades that were evaluated using MDAO. The trades were vehicle wing span, wing chord, and wing material (aluminum vs. composite). Given the volumetric constraints, beyond a certain wing chord, the wing would need to be folded or would require a more complex deployment mechanism (as in the figure). Each of these choices had performance consequences, and was also assigned a complexity impact (aluminum less complex than composite, small chord less complex than a large chord requiring folding).

The resulting performance trades were captured in a MDAO model implemented in ModelCenter, a COTS tool. Notional complexity metrics were used to capture the impact of complexity and implemented in a ModelCenter Figure 42 shows the MDAO analysis expressed as a ModelCenter analysis model, and Figure 44 shows how complexity of the resulting design evolves as performance is optimized.



Name	Value
Model	
total_hp	
Performance	
max_volume	592.3
maxWeight_out	27
maxVolume_out	592.3
other_volume	258
propulsion_volume	105.698
SFC	0.24105
V_cruise	115.613
max_weight	27
other_weight	14.78
wing_weight	2.162
wing_stored_volume	72.8
propulsion_weight	4.68186
fuel_weight	1.37106
actual_weight	22.9949
volume_used	592.3
endurance	5.68786
range	657.589
fuel_volume	155.802
fixed_weight	21.6239
fixed_volume	436.498
hp_fudge	
wing_properties	
wing_type	6
kill_run	0
wing_length	10
wing_weight	2.162
wing_stored_volume	72.8
wing_cost	700
wing_area_total	79.2
other_weight	14.78
other_volume	258
other_reqd_hp	0.35
other_cost	8195
thickness	0.66
chord	6
Sphere	
GeomInfo	

Figure 42 MDAO Analysis Model for Evaluating Performance and Complexity

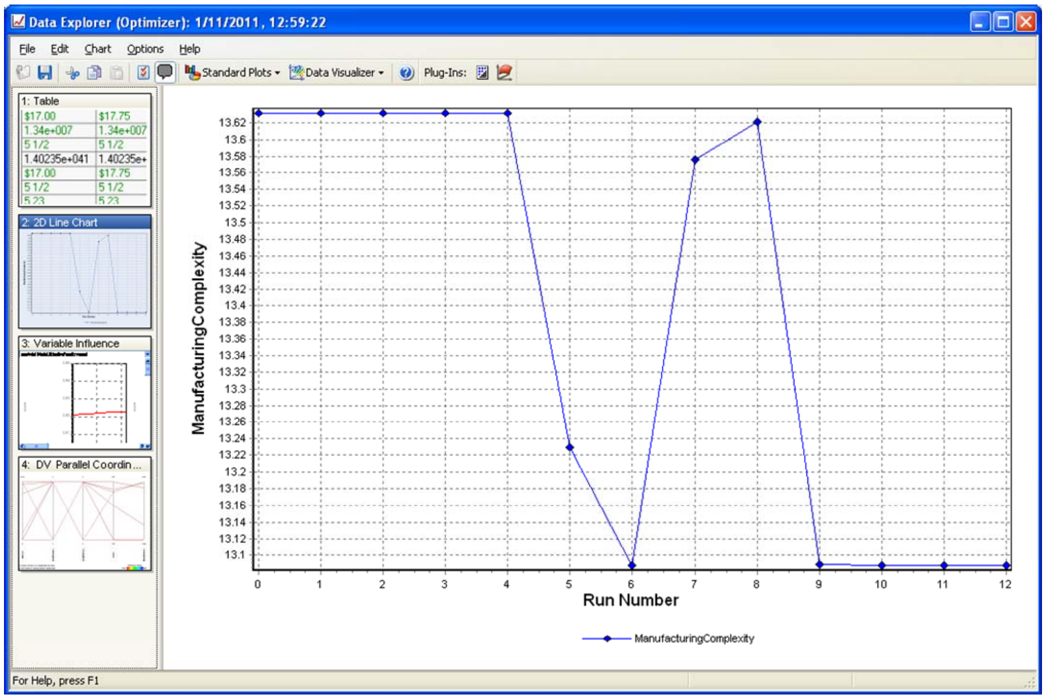


Figure 44 Calculation of Complexity Across a Design Space

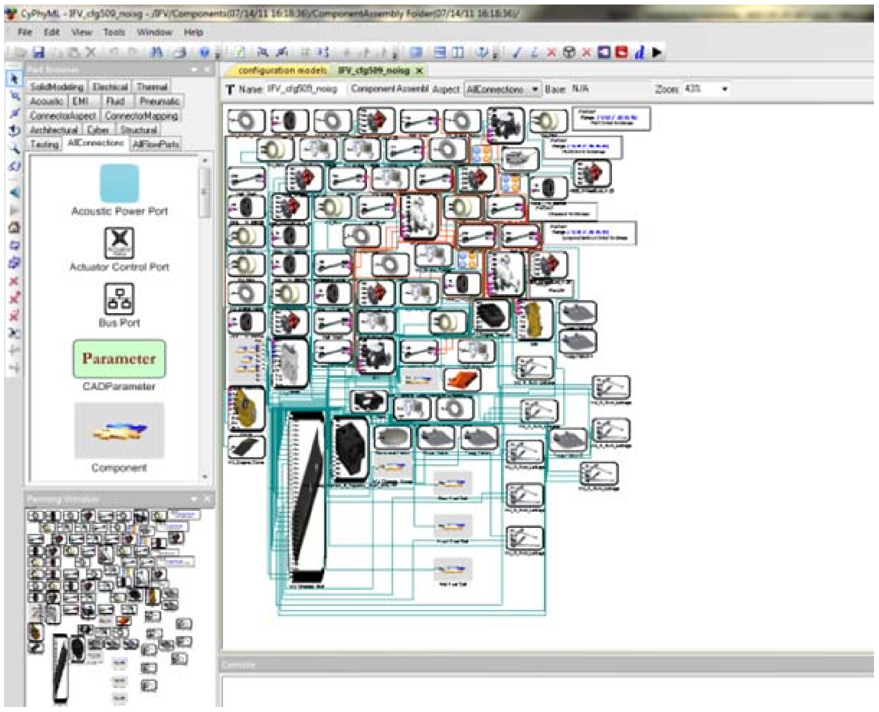


Figure 43 CyPhyML IFV Configuration in GME

Approved for Public Release, Distribution Unlimited.
The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

This experiment validated our basic approach towards the use of the complexity and adaptability metric by stakeholders in making comparative evaluations of candidate designs.

A third integration experiment involved the use of our prototype tool to calculate complexity metrics from CyPhyML models of an IFV challenge problem, through the use of a GME plug-in. GME is the Generic Modeling Environment, a meta-model based model integrated computing tool for defining modeling languages using meta-models, and then generating domain specific tools based on those meta-models. In the case of META, GME was used to develop a meta-model for CyPhyML, the Vanderbilt developed META design language. The Vanderbilt design space exploration tool was used to generate a candidate IFV design expressed as a CyPhyML model (Figure 43). A structural graph of the configuration was then extracted from the configuration model by the prototype tool and used to calculate various complexity metrics. This showed that the tool could be integrated with a META design flow and tool chain and that the information required to calculate complexity metrics could be extracted from the design representation. As part of the experimentation we began with the CyPhy model of the IFV challenge problem design space, and used DESERT to generate 522 feasible vehicle designs. We then generated CyPhy configurations for each of the designs and used the prototype complexity and adaptability tool CyPhy interpreter to extract the directed graphs for the design. The graphs were then passed to the various algorithms that calculate the individual measures that are used to

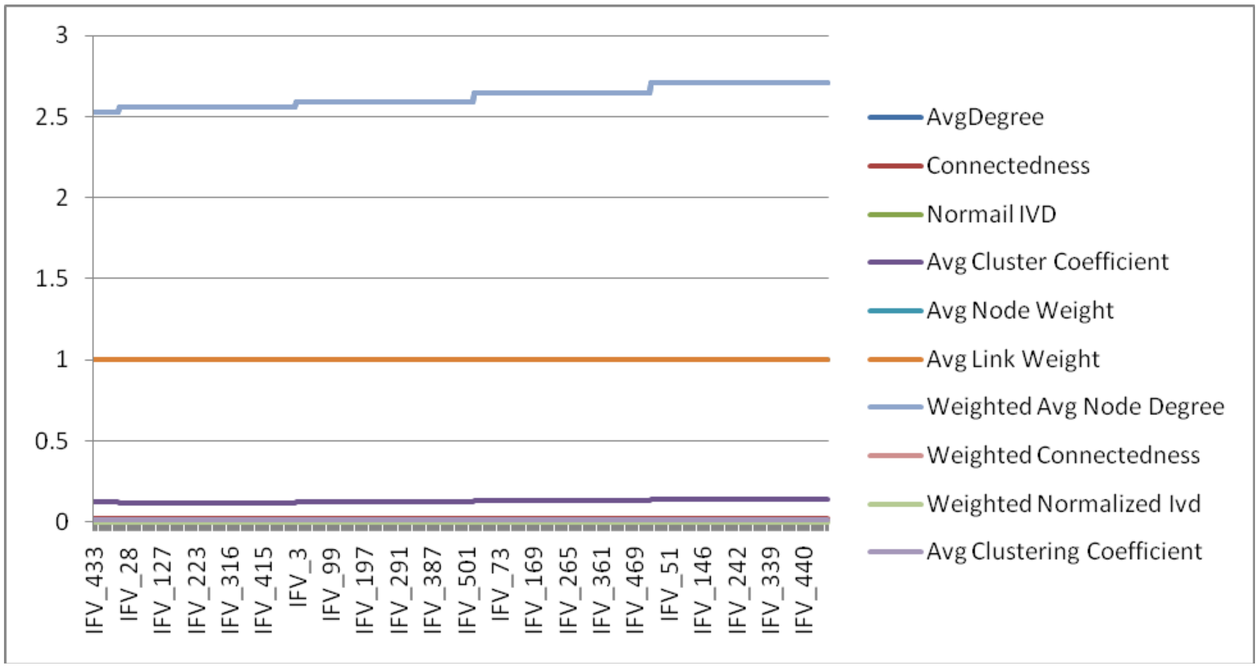


Figure 45 Complexity and Adaptability Measures of the 522 Feasible IFV Configurations

compute the final complexity and adaptability metrics. Figure 45 shows how some of the measures serve to partition the design space into a number of equivalence classes, based on the dominant architectural configurations captured by the design space.

5.0 CONCLUSIONS

Over the course of the META II Complexity and Adaptability project, the contractor sought the answers to four questions: What are the metrics? Do they correlate with programmatic measures such as cost and schedule? Are they a basis for accurate prediction of programmatic data, specifically cost and schedule, or are they akin to past performance disclaimers for a mutual fund – “not necessarily indicative of future results”? Can they be used to evaluate alternate designs? We have developed affirmative answers to all four questions. We have identified a set of metrics that we provide insights into the complexity and adaptability of designs. We have used data from a variety of Boeing programs to identify correlations with cost and schedule, and produce functions that use the metrics to predict cost and schedule. We have implemented these metrics in a prototype tool and applied them to a set of designs produced by META design space exploration tools enabling their use to support design decisions.

We have defined and validated approaches for using the metrics in the META design process. The metrics tool allows the metrics to be calculated for an individual design and presented to the engineer for his or her use. We have developed approaches for integrating the metrics into META design space exploration in two ways. First, we demonstrated the use of the metrics as another analysis technique in multidisciplinary analysis and optimization. Second, in coordination with the Vanderbilt Design Flow team, we develop and demonstrated an approach to use the metrics in conjunction with constraint based design space exploration. In this case, the metrics would be applied to the points in the design space representing viable designs and the results would be presented to stakeholders as part of design space visualization. Each of these approaches enables system designers or other stakeholders to make choices between designs based on metrics results, in addition to all of the other information available characterizing the design. This should enable the stakeholders to make better informed decisions and design choices.

How well the metrics support such choices is a question to which we have provided an initial affirmative answer. We answered that question in two ways. We calibrated the metrics against historical projects in order to establish and quantify a relationship between the complexity and adaptability metrics being calculated on designs and project cost and schedule. Establishing such a correspondence enables stakeholders to make more informed choices between design alternatives by providing them with quantitative insight into the impact of changes in metric values between designs. Quantifying the cost in schedule due to complexity and adaptability for a particular performance gain provides a real context for evaluating the value of incremental performance improvements, or the schedule savings that would result from forgoing a particular capability. We have performed experiments to evaluate the extent to which the metrics we have developed and implemented differentiate between designs, which is another factor in their utility to the AVM and META design process.

6.0 REFERENCES

[1] Braha, D. A. On the Complexity of the Design Synthesis Problem, *28*, 1996.

[2] Maimon, D. B., The Measurement of a Design Structural and Functional Complexity, 28 (4), 1998.

[3] Summers J, Shah J, "Mechanical engineering design complexity metrics: Size, coupling and solvability", J. Mechanical Design, *ASME Transactions*, V132, Feb., 2010.

[4] Suh, N., *Axiomatic Design: Advances and Applications.* NewYork, NY: Oxford University Press, 2001.

[5] El-Haik, B. a. (1999). The Components of Complexity in Engineering Design. *IIE Trans* , 925-934.

[6] Yang, B. E.-H., The components of complexity in engineering design. *31* (925-934), 1999.

[7] Taguchi, G., Robust Technology Development: Bringing Quality Engineering Upstream, ASME Press, NY, 1993.

[8] Hommes Q.D Van Eikemma, "Comparison and application of metrics that define the components modularity in complex products", ASME, IDETC/CIE, Brooklyn, NY, 2008

[9] Ulrich and S. D. Eppinger, "Product Design and Development", 1995.

[10] Hölttä, E.S. Suh, O. de Weck, "Tradeoff between modularity and performance for engineered systems and products", ICED, Melbourne, 2005.

[11] Guo, J.K Gerhenson, "A comparison of modular product design methods based on improvement and iteration", ASME Design Engineering and Technical Conference, Salt Lake City, UT, 2004.

[12] Strong, S.P Magleby, A.R Parkinson, "A classification method to compare modular product concepts", ASME, Design Engineering and Technical Conference, Chicago, IL, 2003.

[13] Bartolomei, J. “Qualitative knowledge construction for engineering systems: extending the design structure matrix methodology in scope and procedure”, Ph.D Thesis, Engineering Systems Division, MIT, 2007.

[14] Newman, M. E. J. and Girvan, M. “Finding and Evaluating Community Structure in Networks, Phys. Rev. E, 2004.

[15] Martin M, Ishii K, “Design for Variety: Development of Complexity Indices and Design Charts”, ASME DFM Conf, Sacramento, CA, Sep. 1997.

[16] Brill P, Mandelbaum M, “On measures of flexibility in manufacturing systems”, Int J Production Research, V27(5), 1989.

[17] Newcomb, B. Bras, D.W Rosen, “Implications of modularity on product design for the life cycle”, ASME Design Engineering and Technical Conference, Irvine, CA, 1996.

[18] Kota, K Sethuraman, R Miller, “A Metric for Evaluating Design Commonality in Product Families”, ASME, Journal of Mechanical Design, Vol 122, pp 403-410, 2000.

[19] Maupin A, L.A Stauffer, “A design tool to help small manufacturers reengineer a product family”, ASME Design Engineering and Technical Conference, Baltimore, MD, 2000.

[20] Chen W, Yuan C, “A Probabilistic-Based Design Model for Achieving Flexibility in Design”, J. Mechanical Design, ASME Transactions, 1998.

[21] Jiao, M.M Tseng, “Customizability analysis in design for mass customization”, Computer Aided Design, V36(8), pp 745-757, 2004.

[22] Simpson T, Rosen D, Allen J, Mistree F, “Metrics For Assessing Design Freedom And Information Certainty In The Early Stages Of Design”, Asme Design Conf, Irvine, Ca, Sep 1996.

[23] Kalligeros, O. de Weck, R. de Neufville, A. Luckins, “Platform identification using Design Structure Matrices”, Sixteenth Annual International Symposium of the International Council on Systems Engineering, 2006.

[24] Shaw G, “Generalized information network analysis for distributed satellite systems”, PhD thesis, MIT, 1999.

[25] Rajan, M.V. Wie, M.I Campbell, K.L Wood, K.N Otto, “An empirical foundation for product flexibility”, Design Studies, Vol 26, Issue 4, pp 405-438, 2004.

[26] Wertz, J. R., and Larson, W. J., *Reducing Space Mission Cost*, Microcosm Press, 1996.

LIST OF ACRONYMS, ABBREVIATIONS, AND SYMBOLS

Acronym	**Description**
ARFL	Air Force Research Laboratory
AFRL/RZPA	AFRL Propulsion Directorate, Power Division, Energy Optimization and Assessment Branch
ASU	Arizona State University
AVM	Adaptive Vehicle Make
BCA	Boeing Commercial Aircraft
BDS	Boeing Defense Space, and Security
CAD	Computer Aided Design
CC	Change Cost
CI	Commonality Index
CI	Integration Complexity
CI_M	Modified Integration Complexity
CM	Coefficient of Modularity
CMEA	Change Modes and Effects Analysis
COM	Component Object Model

Acronym	**Description**
COTS	Commercial Off the Shelf
CPN	Change Potential Number
CPS	Cyber-Physical System
CyPhyML	Cyber-Physical Modeling Language
DAL	Design Automation Lab
DARPA	Defense Advanced Research Projects Agency
DCV	Directional Control Valves
DI	Differentiation Index
DF	Design Freedom
DOE	Design of Experiments
DPI	Design Preference Index
DSM	Design Structure Matrix
DV	Design Variable
FR	Functional Requirement
GME	Generic Modeling Environment (tool)
GMI	Gerhensen's Modularity Index

Acronym	**Description**
IC	Information Certainty
IFV	Infantry Fighting Vehicle
IRM	Interface Reuse Metric
JDBC	Java Database Connectivity
MDAO	Multi-Domain Analysis and Optimization
MDM	Multi-Domain Matrices
PCI	Product Line Commonality Index
PDM	Product Data Manager/Management
PI	Principal Investigator
PoP	Period of Performance
PP	Physical Parts
RMS	Root Mean Squared
RPN	Risk Priority Number
SDB	Small Diameter Bomb
SMI	Singular Value Modularity Index
S/N	Signal to Noise Ratio

Acronym	Description
SUAV	Small Unmanned Aerial Vehicle
SVD	Singular Value Decomposition
UAV	Unmanned Aerial Vehicle
UI	Ulrich modularity index
WBS	Work Breakdown Structure
WI	Whitney index
WPAFB	Wright-Patterson Air Force Base