Spring 2016

A domain independent method to assess system of system meta-architectures using domain specific fuzzy information

Louis Edward Pape II

A DOMAIN INDEPENDENT METHOD TO ASSESS SYSTEM OF SYSTEM META-ARCHITECTURES USING DOMAIN SPECIFIC FUZZY INFORMATION

by

LOUIS EDWARD PAPE II

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

SYSTEMS ENGINEERING

2016

Approved

Dr. Cihan Dagli, Advisor
Dr. Steven Corns
Dr. Ivan Guardiola
Dr. David Enke
Dr. James Paunicka
Dr. Nil Ergin

ABSTRACT

This research proposes a domain independent method to build and assess systems of systems (SoS) architecture models. A simplified binary, meta-architecture containing each component system's participation and a first order, system-to-system interface is proposed. The method describes how to elicit desired SoS attributes from stakeholders. Measures of the attributes depend on systems' participation, characteristics and interfaces, that is, on the SoS architecture. The goal is to model a realizable SoS configuration, optimized over multiple attributes. Key attribute measures are combined in a fuzzy inference system to assess an overall fitness measure for any SoS within the meta-architecture. A genetic algorithm is used to find 'good' SoS architectures with a fitness that depends on the participation framework. This research illustrates a method to define architecture sensitive attributes and build the fuzzy assessor. These are two segments of the Missouri S&T developed, nine part Flexible Intelligent Learning Architectures for SoS (FILA-SoS) research approach to architecting SoS. A desirable SoS architecture found this way may be handed off to an agent-based model to examine the impact of various negotiation behaviors or policies on realization of the SoS. The final configuration may evolve over several development epochs as desired in the wave model.

The method is demonstrated on SoS in several domains to illustrate its broad generality. Two intelligence, surveillance and reconnaissance (ISR) SoS, a search and rescue (SAR) SoS, two versions of the MITRE Toy problem, and a validation using an actual SoS for a large training program are analyzed. The method provides researchers and designers with a novel way to think about the effects of imprecise stakeholder desires, sensitivity to inputs, and acquisition policies on SoS architecting.

ACKNOWLEDGMENTS

I thank my advisor, Dr. Cihan Dagli, for his patient, wise and persistent tutelage as I frequently wandered from the simple path. I would also like to thank the SERC for funding the RT-37, RT-44c and RT-109 research tasks titled "An Advanced Computational Approach To SoS Analysis And Architecting Using Agent-Based Behavioral Modeling." The research colleagues (professors and fellow students) on those tasks were most helpful in refining the approach and providing a forum to expose ideas to scrutiny and implementation tests.

My committee and MS&T faculty and staff were also very helpful with comments and discussions during course work, writing papers, and developing the models and methods discussed in this dissertation. Several committee members also participated in the SERC research tasks, doing double duty for me.

The Boeing Company Learning Together Program paid for the majority of my doctoral classes. I am grateful for that support. The International Council on Systems Engineering (INCOSE) organization, as well as the local Midwest Gateway INCOSE chapter, and my participation in the annual Systems Engineering Research and Complex Adaptive Systems Conferences, were all very helpful with discussions of best practices and their application on potential projects. I am grateful for receiving the INCOSE Foundation/Stevens Doctoral Award in 2013, based on this work while it was in progress.

Finally, I want to thank my wife, Donna, and the rest of my family for their extraordinary patience and support during my extended graduate school journey.

ACKNOWLEDGEMENT OF SUPPORT

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center (UARC) managed by Stevens Institute of Technology.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the United States Department of Defense.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

LIST OF ABBREVIATIONS

A3	Architecture description technique referring to the size paper it uses
AADL	Architecture Analysis and Description Language
ADS-B	Automatic Dependent Surveillance-Broadcast
AF	Air Force
AMADEOS	Architecture for Multi-criticality Agile Dependable Evolutionary Open System-of-Systems
AOC	Air Operations Center
AOR	Area of Responsibility
ARID	Active Reviews for Intermediate Designs
ARIMA	Autoregressive integrated moving average
AS5506B	SAE standard for Architecture Analysis and Description Language
ASD(NII)	Assistant Secretary of Defense for Networks and Information Integration
ASN/RDA	Assistant Secretary of the Navy (Research, Development, and Acquisition)
ATAM	Architecture Tradeoff Analysis Method
ATC	Air Traffic Control
ATM	Air Traffic Managemeent
AV	All View in DoDAF
BLOS	Beyond Line of Sight
BPMN	Business Process Modeling Notation
BS	Bachelor of Science

C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance
CA	California
CC	Closed Circuit
CESUN	Council of Engineering Systems Universities
CIR	Computing Infrastructure Readiness
CJCSI	Chairman of the Joint Chiefs of Staff Instruction
CJCSM	Chairman of the Joint Chiefs of Staff Manual
COD	Criticality of Dependency (in FDNA)
COMPASS	Comprehensive Modelling for Advanced Systems of Systems
CONOPS	Concept of Operations
CONUS	CONtinental United States
CRC	Chemical Rubber Company
CSD	Complex Systems Design
CV	Capability View
DANSE	Designing for Adaptability and evolutioN in System of systems Engineering
DFO	Department of Fisheries and Oceans
DIV	Data & Information View
DoDAF	Department of Defense Architecture Framework
DoDI	Department of Defense Instruction
DSP	Defense Support Program
DYMASOS	Dynamic Management of Physically Coupled Systems of Systems
EO/IR	ElectroOptical/Infrared
EU	European Union

FA	Federated Architecture
FAA	Federal Aviation Administration
FAM	Fuzzy Associative Memory
FDNA	Functional Dependency Network Analysis
FILA-SoS	Flexible Intelligent Learning Architectures for Systems of Systems
FIF	Fund an InterFace
FIS	Fuzzy Inference System
FOP	Fund Operation (of a System)
GA	Genetic Algorithm
GATM	Global Air Traffic Management
GPS	Global Positioning System
GUI	Graphic user interface
I/F	Interface
ICAO	International Civil Aviation Organization
IEA	Information Enterprise Architecture
IED	Improvised Explosive Device
IEEE	Institute of Electrical and Electronics Engineers
IFS	Intuitionistic Fuzzy Sets
INCOSE	International Council On Systems Engineering
IP	It means a funded EU project
IR	Infrared
IS2	Three letter code for the second ISR project
ISO	International Organization for Standardization

ISO/IEC/IEEE	International Organization for Standardization/International Electrotechnical Commission/Institute of Electrical and Electronics Engineers
ISO-10303	Automation systems and integration — Product data representation and exchange
ISR	Intelligence, Surveillance & Reconnaissance
JSTARS	Joint Surveillance and Target Attack Radar System
KM	Karnik–Mendel
KPA	Key Performance Attributes
LML	Life cycle Modeling Language
LOS	Line of Sight
LVC	Live, Virtual, Constructive
MBA	Master of Business Administration
MF	Membership Function
MOA	Memorandum of Agreement
MODAF	Ministry of Defence Architecture Framework
MOO	Multi-Objective Optimization
MOU	Memorandum of Understanding
MS	Master of Science
N/A	Not applicable
NASA	National Aeronautics and Space Administration
NATO	North Atlantic Treaty Organization
NCO	Netcentric Operations
NOAA	National Oceanic and Atmospheric Administration
NR	Net ready
NVS	National Air Space Voice System

OOTW	Operations Other Than War
OUSD (AT&L)	Office of the Undersecretary of Defense for Acquisition, Technology & Logistics
OV	Operational View
PBS	Public broadcasting System
PEM	Program Element Monitor
POR	Program of Record
PSO	Particle Swarm Optimization
RF	Radio Frequency
ROM	Rough Order of Magnitude
RPA	Remotely Piloted Aircraft
RT	Research Task
S2ESC	IEEE Software and Systems Engineering Standards Committee
SAE	Society of Automotive Engineers
SAR	Search and Rescue
SE	Systems Engineering
SEI	Software Engineering Institute
SERC	Systems Engineering Research Center
SESAR	Single European Sky ATM Research
SESARJU	Single European Sky ATM Research Joint Undertaking
SMC	Systems, Man, and Cybernetics
SME	Subject Matter Expert
SOD	Strength of Dependency
SoS	System of Systems
SoS	Systems of Systems
SoSE	Systems of Systems Engineering

SPO	System Program Office
STANAG	Standardization Agreement (NATO)
SWIM	System Wide Information Management
SysML	Systems Modeling Language
TEL	Transporter Erector Launcher
UAV	Unmanned Air Vehicle
UCS	User Control Station
UML	Unified Modeling Language
US	United States
USAF	US Air Force
WG	Working Group

1. INTRODUCTION

1.1 RESEARCH INTO ACKNOWLEDGED SYSTEMS OF SYSTEMS (SoS)

The aims of this research are to develop and explore a model building method that can handle the inherent ambiguities of designing a System of Systems (SoS) comprised of pre-existing, independent systems. The method then uses a fuzzy genetic algorithm (GA) approach to find 'good' SoS compositions among a universe of possibilities. The method itself is domain independent; it is applicable across a wide range of domains with very little tuning required. Missouri University of Science and Technology (Missouri S&T) researchers developed an approach called Flexible Intelligent Learning Architectures for Systems of Systems (FILA-SoS), comprised of nine segments to explore the SoS design problem space. This research makes up two of the nine segments of FILA-SoS shown in Figure 1.1. A secondary goal of the method is to increase the understanding of relevant trade-space issues and possibilities for modeling components and capabilities in the design of SoS under multiple objectives from an acquisition viewpoint. FILA-SoS starts with a simplified, binary meta-architecture for the participation of each potential system, and the presence of each system's interface with every other system. The method of generating and assessing SoS designs within this meta-architecture comprises the following steps:

- Developing an SoS concept, nominating potential systems, and collecting domain data
- Eliciting desired SoS attributes and their relative values from the stakeholders
- Hypothesizing, documenting, and implementing algorithms (models) for evaluating each attribute from the SoS meta-architecture

- Finding a rule based combination of attribute values for an overall SoS assessment, or fitness, through a fuzzy inference system
- Checking the attribute models against the SoS meta-architecture to ensure closure (which may cause a repeat of previous steps as far back as step two if the checks are not satisfactory)
- Selecting a satisfactory architecture with the fuzzy genetic algorithm (GA)
- After other segments of the FILA-SoS approach find a potentially sub-optimal 'realizable' and agreed to design through negotiations, the fuzzy assessor is exercised again to provide a measure of the fitness of the final architecture for that epoch, or wave, in the wave model of SoS evolution (Dahmann, et al. 2011).

The method is demonstrated on several hypothetical SoS tuned to show the feasibility of the approach in general, on variations of a classic MITRE 'Toy' SoS using functional dependency network analysis (FDNA) as a different problem formulation, and on a large, customer provided (proprietary) existing live, virtual, constructive (LVC) training SoS for validation on a real-world example.

1.2 SOCIOTECHNICAL SYSTEM COMPLEXITY

There can be no question that today's civilization and its component systems are far more complex than in previous times (Wai 2012). Travel, trade, commerce, education, technology, nation states, financial networks, populations, legal frameworks, volume of information, magnitude of risk, political systems, interconnectedness and interdependency – all these facets of society have expanded tremendously in scope and

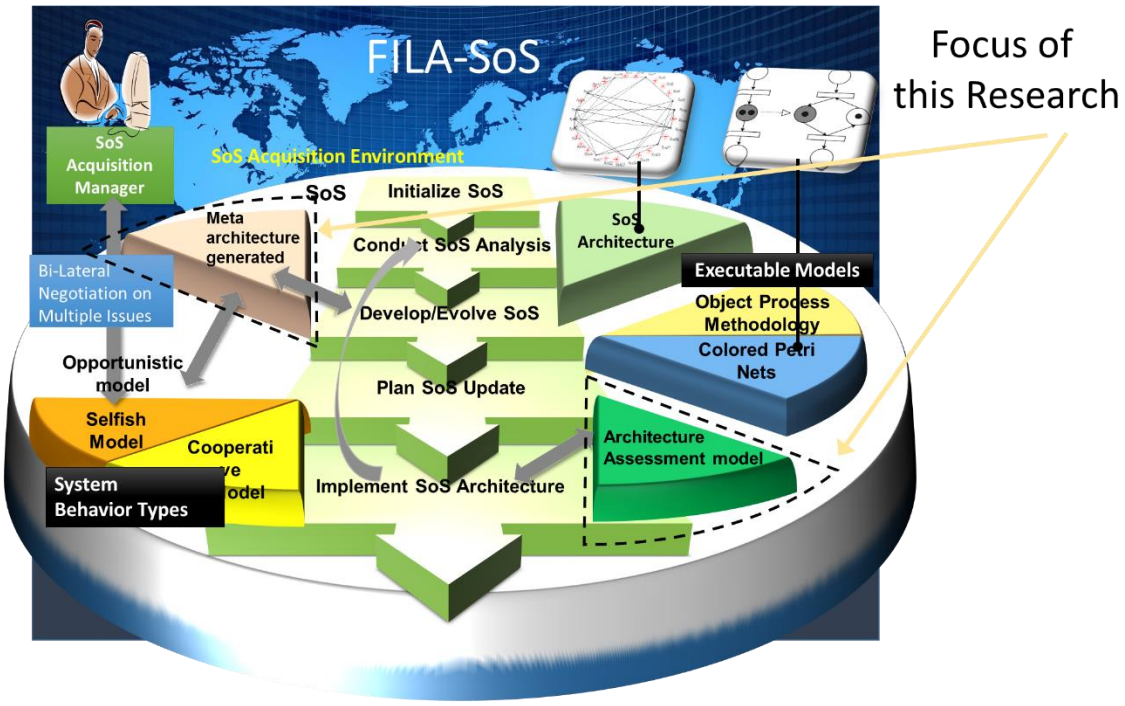


Figure 1.1. Focus: two segments of the FILA-SoS approach

reach over recent centuries. The systems that allow, or control (depending on the viewpoint), these facets of civilization are called sociotechnical systems. It is only recently, however, that humanity's systems have become so powerful and interconnected that we can no longer afford avoidable mistakes. Society's ability to know, and to do, more and more have almost kept pace with its desires. There have always been problems associated with growth; there have always been unintended consequences of decisions, even with the best intentions and significant care by decision makers. The downside risk inherent in the most powerful societal systems has grown too large. One need look no further than nation-states' nuclear weapons establishments, the recent worldwide financial collapse, the too numerous environmental disasters, or widespread ethnic cleansing to find examples of this downside risk. Almost every large system is now a

sociotechnical system, meaning that both human and technological aspects are deeply entwined. This implies that they are more difficult to analyze as well. Almost every large socio-technical system is now both complex and adaptive, meaning that results:

- Are not always predictable,
- Could be strongly influenced by small perturbations, and
- Can develop in ways neither contemplated by designers nor understood by users.

Further, many of these societal systems are in fact Systems of Systems (SoS). Systems of Systems Engineering (SoSE) is becoming a significant area of specialization within the profession of Systems Engineering (SE). Changes inevitably occur in society's institutions, technologies, governments, and patterns of life; other changes inevitably need to be made to accommodate the first changes. Society's ability to analyze SoS, and to understand the implications of change, needs to improve beyond current practice to avoid costly mistakes and errors that might realize the downside risks mentioned above.

Complex SoS structures, with divergent stakeholder viewpoints on what constitutes success and multiple, frequently contradictory, objectives, are the norm in large, modern socio-technical enterprises. These SoS will be expected to do things never attempted before; to be safe, effective, efficient, to have little environmental impact, to be easy to understand, to never fail, to work in conditions far removed from those for which they were designed, and above all, to be inexpensive to build and operate.

An acknowledged SoS is any simultaneously semi-voluntary and partially regulated combination of systems with a centralized goal, but a less than complete central

authority. These include: multi-jurisdictional construction projects (canals, tunnels, bridges or dams), non-governmental organization (NGO) relief efforts, airports, seaports, multimodal transportation systems, security architectures (both physical and cyber), supply chains, and health care as an enterprise. The method described here can be used as a starting point for understanding some of the possible trade space in acknowledged, DoD style SoS, as well as many non-military, complex, multi-stakeholder SoS constructs. Society requires the ability to better understand how acknowledged SoS develop, evolve, and thrive, so as to better manage organizations, resources, and change in the future.

1.3 NEED FOR IMPROVED METHODS OF ARCHITECTING

As civilization grows, the need for larger, more complex systems also grows. Many of the newest, most complex systems are better described as systems of systems (SoS), in which existing, independently developed and managed major systems are brought together to achieve additional capabilities not possible through the component systems' continued independence. Examples of this include

- Ballistic missile defense, in which existing warning radars, communication systems, and shelters are combined with new technology interceptor and decision systems
- A modern multimodal transportation system would certainly qualify as an SoS – not merely an interstate highway system, for example; but also feeder roads; airports; seaports; tugs; canals; rivers; barge, rail, trucking, bus, and cab companies; automobiles; aircraft; ships; warehousing; hotels; rest stops; travel and liability insurance; fuel and repair stations; traffic laws, courts, taxes, tolls, customs, tariffs, and so on

- Governments, the internet, the world economic system and multinational, conglomerate corporations may be characterized as systems of systems to varying degrees, and with varying ranges of central control.

New possibilities from new technologies, as well as from new ideas about ways to use existing technologies and systems, allow society to pursue many paths not previously discoverable. However, the growth of possibilities is currently outstripping the availability of resources, even as fast as the availability of resources within the world economy has grown in the last couple of centuries. Examples of this include:

- The US Apollo program of the 1960's, had access to nearly unlimited resources and reached the moon, but the US would have difficulty duplicating the feat today due to competing priorities.
- The Space Shuttle, developed in the 1970s, operated as the nation's space transportation system for the next three decades with enormous technical success over hundreds of flights (except the two disasters and cost overruns). However, the replacement program decision was postponed repeatedly until after the shuttle was retired, leaving a gap in launch capability. NASA's budget has barely kept pace with inflation for the last 20 years.
- 'Big physics' provides another example of decline. Another program that started with nearly unlimited resources in the Manhattan Project, followed by larger and larger particle accelerators. This led to a better understanding of deep physics and collaboration with astrophysics. The eventual rejection of the proposal for the U.S. Superconducting Super Collider in the 1990s as

being too expensive for an unknowable return (in addition to environmental concerns) is a sign of society's inability to continue down that promising path.

- Very large tunnels, such as the Channel Tunnel, the Sendai Tunnel, or the "Big Dig" in Boston qualify as projects started in an era of relative abundance (1980s and 1990s); they all resulted in huge overruns and repeated bankruptcies of the sponsor corporations. These were all large, complex projects involving many investors, governmental jurisdictions, government agencies, subcontractors, specialists, and regulators working as an SoS.

None of these projects, nor numerous other proposed large construction projects, could possibly get started in today's economic environment. Author and futurist Neal Stephenson famously bemoaned "our far broader inability as a society to execute on the big stuff" (Stephenson 2011).

Very large projects, invariably SoS, not only are expensive and difficult to manage by definition, but almost invariably overrun in cost and schedule. While one reason might be that if reasonable (instead of over optimistic) cost and schedule estimates were originally presented, no one would ever choose to start these projects. It is generally regarded as self-defeating to acknowledge realistic costs when trying to get a large project started. However, if the proponents do get a project started, it is highly likely that it will contend with many problems and repeated upward revisions to the estimate to completion. Although business schools teach that sunk costs are not a reason to continue investing in a project (Krugman and Wells 2009), the emotional attachment to the sunk cost is practically impossible to ignore. In society's defense, it is seldom the case that it would be cost effective to start a large project over from the beginning.

Another compelling reason for delays and cost overruns is that larger projects typically have an impact on larger numbers of stakeholders. They have a 'bigger footprint,' i.e., they may benefit many stakeholders, but they likely also impinge on numerous stakeholders in a negative way, leading to strong reasons for obstructionism by larger numbers of opponents. Note: the reason for starting an SoS is to do some new task, or an old task in a far better way. This means changes to the pre-existing way of doing things; hopefully, big changes! Woodrow Wilson once said, "If you want to make enemies, try to change something" (Shaw 1918). Smaller projects are easier to manage in newer and more efficient ways, because less of everything is at risk. But big projects are what society frequently needs. In addition to increasing societal desires over time, numerous other rapidly increasing factors are nevertheless quite compelling:

- Public expectations of being able to do better today than ever before
- Demands from customers and stakeholders to do more with less
- Population growth requires additional social services, such as food, water, energy, transportation, sewage, and sanitation
- The apparent growth of the potential of technology, systems and networks
- Rapidity of technological change
- Global competition
- Threats, whether 'simply' competitive, or security/existential in origin

While these factors tempt society to pursue larger projects with greater goals, there are factors that oppose:

- Limitations of resources, especially when the projects are large

- The ‘cost of regret’ for projects forgone, when significant resources are committed to one project over other potentially desirable projects
- Lack of large numbers of fully trained personnel to manage development and operate the otherwise realizable systems
- A changing regulatory environment
- Rapidly rising public demand for less risk, increased safety, and zero environmental impact
- Inability to accurately predict performance, schedule or costs in the face of complexity and uncertainty
- Inability to test all (or even sufficient) combinations of inputs and environments on the operation of complex systems
- Complexity (non-intuitiveness and unpredictability) of the effects of interconnectedness of society’s institutions

Civilization has reached a point where there is a significant need to develop better ways to envision, and to evaluate future possibilities before devoting substantial resources to potential solutions. When the decision is to proceed, one might expect a wise civilization to choose more efficient utilization of existing but finite resources, whether they be capital, work force, time, natural resources, or political support. The trend demands that society architect its possibilities with improved understanding over what was available in the past. Better tools must be combined with improvements in modeling techniques to make this possible.

1.4 ACKNOWLEDGED SoS

Very few SoS are actually under the 'tight central control' typically attributed to military organizations by nonmilitary members. On the continuum of 'degree of central control' of SoS described in the SE Guide for SoS, ranging from extremely tight to near anarchy, almost no SoS exist at either of the far ends of the scale (Director Systems and Software Engineering, OUSD (AT&L) 2008). Most SoS exist near the center of the scale, as *acknowledged* SoS; where there is some recognized central authority, but not complete, centralized control, authority, or budget. Even in the military, authority is broadly delegated. Staff coordination among nearly autonomous functional areas is strongly enforced, implying that most serious decisions are through consensus.

The definition of an *acknowledged SoS* is an overlay on existing component systems that have independent existence outside the proposed SoS. Components of SoS are usually 'legacy' systems, having their own well developed architectures, missions, stakeholders, and funding sources (Bergey 2009). Moreover, successful managers of acknowledged SoS understand that their potential component systems work best if they are perturbed as little as possible to meet the new requirements necessary to contribute to the incipient SoS. That is, the component systems' architectures are primarily left to the systems engineering and architecture professionals at the systems' hierarchy level. It is in the best interest of the SoS manager to coordinate and guide individual systems to join the SoS team, rather than attempt to issue commands or demands.

On one hand, the component (existing, independently managed) systems have no need to accede to an acknowledged SoS manager's requests/demands, nor to officially report through SoS management staff teams. On the other hand, there may be numerous reasons to cooperate with the SoS manager's desired changes to their systems. These

include the opportunity (or excuse) to break open their architecture to make those minor adjustments required to join the SoS – this could allow an opportunity to fix some ongoing problems that do not, on their own merit, justify such 'breakage.' Another reason might be to stretch out the life of the program (and its constituency of stakeholders and contractors) with fresh, new tasking, when the system would otherwise be approaching its end of life, decommissioning, disposal and end of the program office's life. A system might already be planning to make changes to its architecture that could easily accommodate the desired SoS changes, but the new SoS opportunity could be a bonus source of funds or the basis of further upgrades to make the system more relevant within its own domain.

There is no assumption that existing missions of the component systems would not suffer from a decision to participate in the SoS. The most general case is that there *will* be negative impacts to existing missions. In spite of this, there are many potential ways to persuade a system to participate. The nature of these ways may change depending on where each system currently is within its life cycle. Sometimes the small changes required in a system to be able to participate in an SoS may also improve its ability to perform its existing missions. The SoS manager typically has a small budget to help make these changes, so they can be implemented at no net cost to the system. Sometimes there is a backlog of changes planned for the next upgrade of the system; however, without an impetus such as the need for a change to accommodate the SoS, the system is reluctant to initiate the implementation of those minor upgrades. Some reasons for this may be that it is embarrassing to acknowledge the need for a change in a deployed system, the changes are planned (and budgeted) to occur in a specific sequence,

or the proposed funding and schedule seem unrealistic. On the other hand, the system program office (SPO) may be eager for an excuse to 'piggyback' some of its more critical backlog of changes on an outside request from the SoS. A system early in its deployment may welcome an excuse to add a change to its baseline, if only to give it more time to meet its own requirements, if not also providing justification for minor changes of its own as well. A system late in its lifecycle may welcome the chance to stay relevant longer by joining the new SoS. Many times, the minor changes to accommodate a new interface can make an existing system more flexible, or even improve its legacy mission capabilities, but those changes were not judged worthy on their own merits. Madni discusses both the pros and cons of increasing interoperability (Madni and Sievers 2014). It is possible that the overarching SoS mission is important enough that stakeholders of the individual systems' missions agree to the minor degradation to their systems' capability to be able to contribute to the SoS capability. Finally, in the type of acknowledged SoS under discussion, a SPO is free to refuse to participate if the potential degradation to its current capabilities would be undesirable or unjustified by the value of the successful SoS, or not reimbursed enough for making changes necessary to join the SoS.

The above noted issues about impacts to legacy systems are adjudicated during the negotiation phase of FILA-SoS. Motives for the systems' behavior and environment may be modeled there as well. The optimization process may reject participation of a system or interface because its presence diminishes the evaluation of at least one of the attributes. During the architecture planning phase, the SoS manager has only estimates of

cost, performance, and other input data to calculate the attribute evaluations. After a first draft of the SoS model is created and tuned to provide potentially good SoS designs,

- The GA finds a good SoS architecture chromosome
- The SoS design is handed over to negotiation where the domain input data may be adjusted
- The input domain data is improved with negotiated values (including the effects of degradation (or improvement) to individual systems' capabilities from required minor changes to participate and interface with the remainder of the SoS.
- A better, validated model of the system's contribution to the SoS attributes may be used to re-evaluate the selected or negotiated architecture.

1.5 THE SoS ENVIRONMENT

This modeling framework is offered for an acknowledged SoS, where each component system is a fully functioning, independently funded and managed system represented by the (SPO) that manages the program. A high-level (SoS level) official envisions an opportunity to achieve a needed, new capability by using combinations of existing systems in a new way such that component systems can be left largely unchanged, or incorporated with relatively minor changes. The acknowledged SoS approach is only useful if it can achieve the new capability under either or both of the following constraints:

- A reduced cost compared to a separate, new 'purpose built' system, and/or
- A reduced time to field such a new capability.

Defense Secretary Rumsfeld famously said "…you go to war with the army you have, not the army you might want or wish to have…" (Suarez 2004). Therefore, the concept of the acknowledged SoS meta-architecture is that the major capabilities are built into the systems already, but small, quick changes can be made to interfaces to enhance existing capabilities when used in a cooperative manner. The proposed method uses a novel, binary system and interface architecture, that will be called the meta-architecture throughout this document. It will guide the SoS architecture development through a wave model evolution in capabilities over time (Dahmann, et al. 2011) with incremental improvements after it begins operation.

The new capabilities being sought in the SoS are achieved by combining mostly existing system capabilities and/or adding new capabilities that arise in conjunction with other systems (i.e., through new interfaces) (CJCSI 6212.01F 12 Mar 2012). If simply throwing more systems (with their individual capabilities) at the problem were sufficient, there would be no need to create the SoS. Therefore, all successful acknowledged SoS architectures need to invest in the relationships (i.e., interfaces) between the systems composing the SoS. Furthermore, improvements in typical SoS attributes such as performance, availability, affordability, reliability, etc., must arise from the interfaces. Otherwise, there is no advantage over simply adding individual systems' capabilities. The nature of the acknowledged SoS implies that the SoS manager does not have absolute authority to command system participation (nor interoperability changes). Instead, she must 'purchase' the component systems' participation and modifications, not merely with funding but also through persuasion, the strength of the vision of the SoS, quid pro quos, the bully pulpit, appeals to good sense, and whatever other means are

legitimate and effective (Director Systems and Software Engineering, OUSD (AT&L) 2008). Individual systems remain free to decide not to participate in the SoS, although that choice may cost those systems something as well. That cost comes not only from the withholding of SoS funds, but also as the missed opportunity to participate in a successful SoS, missed opportunity to make a related change, or as earning a reputation for uncooperativeness for the common good.

Additionally, some of the desired systems may not be available to the SoS during a particular operational period of need even though they made required interface changes. They may be down for maintenance, assigned to a higher priority mission, or geographically distant on their day-to-day missions and therefore, not able to contribute. Some of the required capabilities and interfaces may already exist in the systems, meaning they are free and fast for development, but those systems may have a significant cost to operate in a fielded SoS. This may be a reason term of art school. All right to ask them to participate. Some systems may have enough capability that the SoS can tap their spare capability while they pursue their original tasks, so they are essentially free to operate. Other capabilities may need minor (compared to a 'new start' major program) development efforts, either within the system or by developing a new interface with another system. The performance capabilities of the SoS will generally be greater than the sum of the capabilities of its parts (Singh and Dagli 2009). If this were not the case, there would be no need for the SoS. Changing the way the systems interact, i.e., tactics alone, with no physical modifications, typically would not improve the SoS capabilities as much as providing completely new ways of interacting through new interfaces. It is assumed that tactics changes do not require an SoS approach. Systems architecting in the

overall context of the SoS must address all the attributes of groups of disparate systems as well as crucial issues affecting their collective behavior.

An instance of an acknowledged SoS might be a military command and control SoS that has transitioned from a tightly knit group of a few systems to an acknowledged SoS that now includes many more previously independent systems. This could be due to a change in the implementation or importance of the missions currently being supported, or of a change of importance and increase in complexities of potential cross-cutting (new) SoS capabilities (Dahmann, Baldwin and Rebovich 2009). Another acknowledged SoS might be a regional air traffic control (ATC) system that crosses national boundaries. National ATCs are independent, but find it strongly in their interest to cooperate and interface with the regional ATC.

One way to develop better tools for predicting performance in various attributes is to use proposed new tools on a very simple model, where the results can be calculated independently. Exploring the working of a tool on simple models can build confidence that the tool does what it is intended to do. Another way to build confidence is to choose a model that can be extended in a very straightforward manner to more complex situations. Actual SoS may have very complex architectures, but at the most basic level, they may be boiled down to '*are the systems here* or not, and *which of them interface with each othe*r.' If they do not interface with each other, they are not an SoS, but simply a collection of systems. This simple, generic model of the SoS is the basis of the FILA-SoS meta-architecture.

1.6 THE SoS META-ARCHITECTURE

A meta-architecture is an organization or pattern by which other architectures may be described. The SoS meta-architecture for this analysis consists of a list of all the potential component systems, followed by the first order interfaces of each system with every other system. Associated with each of these labeled elements is a single bit representing presence (1) or absence (0) in a particular architecture instance. The meta-architecture is the empty framework, or bit string, with the positions identified as to their meaning. An instance of the meta-architecture occurs when the framework is filled with ones and zeroes. An instance of the meta-architecture is also called simply an architecture or an SoS. One binary bit indicates the presence of a system, other binary bits indicate the presence of the interfaces between that system and each other system. The string of bits representing an architecture is used later in a genetic algorithm, where each string is called a chromosome. An instance of the meta-architecture is a particular arrangement of the ones and zeroes in the string of bits; it is a particular architecture showing which systems with their interfaces are participating in the SoS. The terms architecture, instance, chromosome and SoS are used interchangeably to represent a particular design of an SoS as discussed further below.

The interfaces are assumed to be bidirectional for simplicity; an interface of system i with system j is the same as the interface from system j to system i. Furthermore, in this ‘participation architecture’ – the presence of the system is equivalent to the decision by the SPO to participate and is represented as a ‘1’ in the architecture. The decision by the system (used interchangeably with the term SPO) to have an interface with another system is also represented by a ‘1.’ If the system or interface is not present (not participating), it is represented by a zero.

Figure 1.2 and Figure 1.3 show an SoS architecture as a long string of bits (X_i, where X is a one or zero) where the position determines which element is indicated. There are **m(m-1)/2** interfaces for an SoS with *m* systems, plus the **m** systems themselves, so the total number of bits in the meta-architecture with **m** systems is **m(m+1)/2**. The meta-architecture consists of all possible bit strings of this length.

X_1	X_2	X_i	…	X_m	$X_{1 \text{ with } 2}$	$X_{1 \text{ with } 3}$	$X_{1 \text{ with m}}$	$X_{2 \text{ with } 3}$	…	$X_{i \text{ with j}}$	…	$X_{(m-1) \text{ with m}}$
Systems					Interfaces							

Figure 1.2. Linear representation of the generalized SoS meta-architecture

Systems																					Interfaces to System 1																				
S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	i1-2	i1-3	i1-4	i1-5	i1-6	i1-7	i1-8	i1-9	i1-10	i1-11	i1-12	i1-13	i1-14	i1-15	i1-16	i1-17	i1-18	i1-19	i1-20	i1-21	i1-22
1	0	1	0	0	1	0	1	1	0	0	1	1	1	0	0	0	1	0	1	0	0	0	0	1	1	1	0	0	1	1	1	0	1	1	1	0	0	1	1	0	1	1

Figure 1.3. Partial linear display of an SoS chromosome extending far to the right

The linear representation of the chromosome representing one instance of an architecture as shown in Figure 1.2 or Figure 1.3 is relatively cumbersome. It is difficult to decide what any particular bit represents without extensive counting, labeling, or other efforts to keep track of it. An alternative representation of the chromosome was found to be an upper triangular matrix. The advantage of the form shown in Figure 1.4 is that the interfaces may be identified ‘by their conventional matrix element row and column position labeling. This form of representation is close to, but not the same as, what is sometimes called an adjacency matrix. The interfaces could be considered a non-directed graph where the nodes are the systems. Usually, the diagonal of the adjacency matrix

would be zeroes, but there are advantages to putting the systems on the diagonal in this representation, so it is not quite the same as an adjacency matrix.

The FILA-SoS meta-architecture allows the representation of many network architectures. The well-known star, ring, fully-connected mesh, partly-connected mesh, and hierarchical 'branch and leaf' network connection topologies are shown with a corresponding representation within the meta-architecture in Figure 1.5. Ones in interface regions of the matrices are shaded, zeroes are not. The second row of matrices shows the effect of numbering the nodes from a different starting position or in a different sequence but they are equivalent from an meta-architecture point of view.

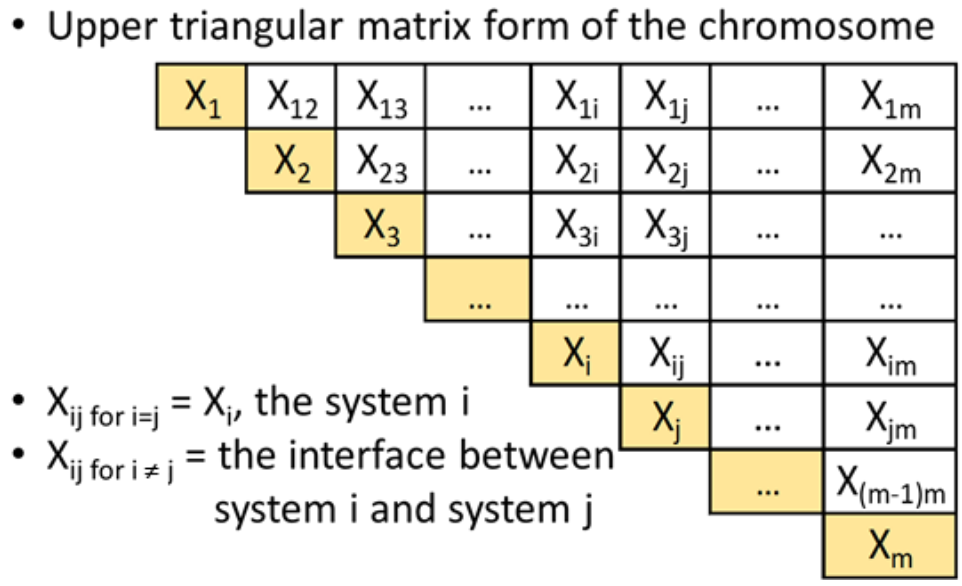


Figure 1.4. SoS meta-architecture layout

The proposed meta-architecture framework may be used to represent any acknowledged SoS. All candidate component systems are represented along the

diagonal, and all potential undirected interfaces are represented in the elements in the upper triangular matrix above the diagonal. Large numbers of component systems and interfaces may need to be examined in designing a typical SoS. A system or an interface may be *excluded* by fixing a zero in the appropriate place in the matrix. Systems or interfaces may be *required* by fixing a one. In the GA approach, most or all of the elements may take on either a one or a zero.

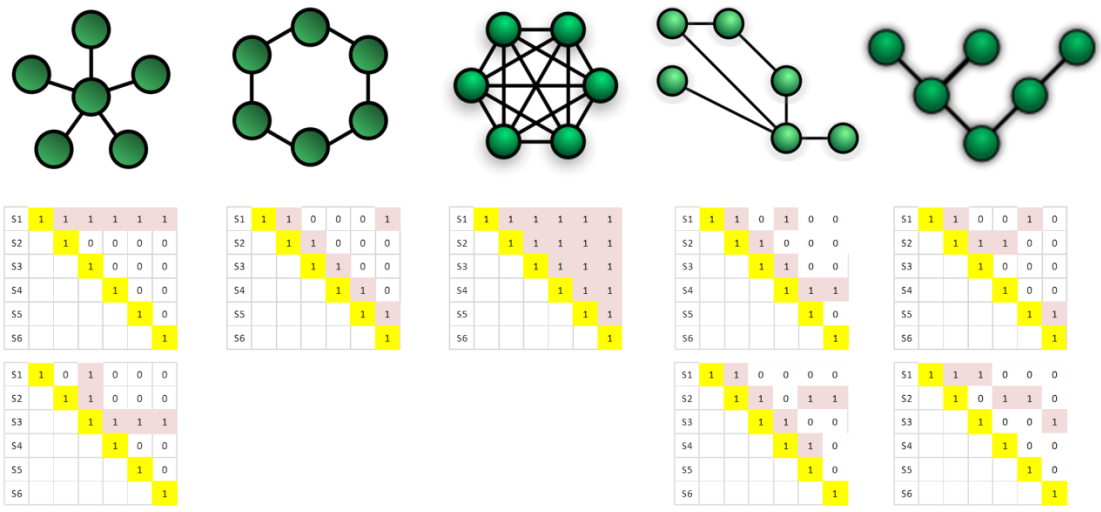


Figure 1.5. Network connection topologies shown in upper triangular form

Resource availability may limit the installation of interfaces by their cost (whether measured in money, downtime, weight, drag, etc.), or limit the use of interfaces by restricting their bandwidth, detectability, or power consumption for example. Those are only a few of the most easily imagined limitations. In choosing appropriate attributes to measure the SoS value proposition, one must consider the most important or significant limitations in the algorithms if they depend on the SoS architecture. The problem in

designing an SoS is to select from the very large range of possibilities while simultaneously trading off among the numerous, important criteria that participants and sponsors need to be satisfied about to support the SoS. Finding the balance of how many considerations to count while keeping the algorithms simple enough to understand and explain, is an art. There may also be potential goodness in some new interfaces outside the existing ones in an imagined SoS – the systems might open themselves up to accomplishing other missions more effectively, either alone or in another SoS to which they contribute some of their capabilities. It is the facilitator's task to ferret these possibilities out of stakeholder and subject matter expert (SME) interviews. These types of issues certainly impact the cooperativeness of a system when negotiating its joining this SoS.

The solution approach must aid the understanding of the impacts of tradeoffs among the various elements and attributes of the SoS. The solution must also account for the behavior of the individual component systems and their motivations in negotiating to participate in, and contribute to, the SoS. Many stakeholders, each with their own system's day-to-day as well as strategic, management issues, are involved with the issues that affect these decisions. Some stakeholders care about multiple systems or even larger SoS issues. They naturally have at least slightly different perceptions of what is important, and even the definitions of the terms used to describe the attributes of the SoS, their own systems, and others. One way to handle the ambiguous linguistic terms commonly used by the stakeholders to describe their needs and wants is to use fuzzy logic.

A partial membership function overlap is one way to handle the uncertainties at the edges and overlaps of these ambiguous usages. Fuzzy approaches are often used in decision support problems (Pedrycz, Ekel and Parreiras 2011), but have not previously been widely used in SoS architecting. Commercial architecting tools such as Core, Sparx, MagicDraw, Rhapsody, or Aris, working in the unified modeling language (UML), systems modeling language (SysML), or business process model and notation (BPMN), for example, do not generate alternative architectures, but the system description and data to be modeled must be provided to them (Hunt, Lipsman and Rosenberg 2001) (Sumathi and Surekha 2010). The proposed method provides an approach to meeting many of the ambiguity and uncertainty concerns for the variety of architectures, key performance attributes (KPAs), and stakeholders possible within an SoS. In this method, the KPAs, which make up the evaluation criteria of the SoS, are defined in terms of the meta-architecture of possible combinations of systems and interfaces. The method must hypothesize an algorithm for each attribute using the chromosome and information about the selected systems and interfaces to produce an SoS attribute evaluation from the system/interface participation. The various attribute algorithms are explained and vetted among the stakeholders to reach consensus on their definition. The rules for combining KPA evaluations to arrive at the SoS assessment are discovered, explained, and vetted through interaction with the stakeholders in the same way. At the end of the method, all stakeholders should understand how the model works and how architectures are assessed.

There is a need for an approach to handle the ambiguities in the selection of the SoS design based on consensus on the KPAs quality assessments over the majority of

their range. Few disagreements occur for the very, very bad or the very, very good assessments. Disagreements typically occur at the edges of the granularity regions. This is an excellent application of the partial membership function principle of fuzzy logic. Some people think a particular KPA value is very bad; others think it is simply far below par, or perhaps only at the low end of average. The solution: let that point have proportionate membership in each evaluation.

An advantage of representing the SoS in the form of a binary string is that the chromosome may be used in a genetic algorithm (GA) approach to explore the evaluations of various instances of the SoS architecture. Optimization might be a bit too strong of a word for what the genetic algorithm can do in this case. Due to the multiple layers of uncertainty in:

- The cost and performance estimates for various aspects of the systems
- The truncated binary (fully present or completely absent) nature of the model
- The simplifications inherent in the high level of abstraction used in the KPA algorithms.

The GA approach primarily helps one explore, in an unbiased way, the influence of rule or component changes. Some KPAs of the SoS remain ambiguous even after extensive discussions among the stakeholders. Fuzzy logic approaches may be used to compare relative scores among many attributes, criteria, and alternatives through algorithms using the presence or absence of the systems and interfaces as input. If an attribute cannot be described in such a way that it depends on the meta-architecture, then

it may not be useful in describing the value of the makeup and organization of the SoS as represented by the meta-architecture.

Finally, there is an inherent difficulty simply in the size of the mass of data about the systems, interfaces, attributes, and the resultant desired versus delivered SoS. It is difficult to comprehend and analyze this mass of data for even one, much less for many, proposed SoS architecture alternative. The modular fuzzy genetic approach proposed here allows simplified models to be used to explore relationships and improve understanding so that one can know where the benefit of improving the fidelity (and possibly the complexity) of individual attribute models and rule sets lies in future efforts.

1.7 AIMS OF THIS RESEARCH

The aims of this research are to develop, document, refine, explore, and demonstrate a method for planning, coordinating, assessing and building successful, acknowledged SoS. The overall approach offers a new way of thinking about many design issues for SoS by combining numerous simple models in a meta-architecture framework. The application of fuzzy genetic algorithms in the SoS wave model acquisition environment makes several difficult areas more tractable. The research should help practitioners quantify potential gains from netcentric interoperability, evaluate SoS lifecycle costs, and explore the impact of high-level policies on SoS concepts.

1.8 PROPOSED MODELING APPROACH

Since acknowledged SoS are typically complex, with multiple stakeholders and continuing missions for the component systems, a multi-objective optimization (MOO)

approach is used to recommend an architecture from the meta-architecture framework. A fuzzy genetic approach is one form of MOO that may be applied in the creation and analysis phases of an acknowledged SoS development over a wide range of problem domains. This approach lends itself to handling the evolution of an SoS over multiple epochs as proposed in the wave model, which is currently a problem of great interest (Dahmann, et al. 2011).

The architecture selected by the GA may be used to begin negotiations between the SoS manager and the selected component systems' managers to find a realizable SoS architecture. In the next epoch of the wave model, the solution may be further developed to evolve the design of the SoS. The current state of the art in system of systems engineering (SoSE), fuzzy linguistic analysis, multi-objective optimization and the gaps that this research fills are detailed in the literature review in chapter 2.

The method is a decision making aid for the SoS manager. It does not so much find the best solution to designing an SoS as help the manager explore the influence of the various constraints on the shape of a reasonable solution. The method starts, as shown in Figure 1.6, from the SoS context and goals using the simplified binary meta-architecture, including the full range of candidate systems and their interfaces. Guided interviews uncover the SoS purpose, characteristics of candidate systems, key attributes that characterize the SoS, and methods for measuring the SoS in each of these attributes. The key attributes generally lend themselves to linguistic characterization and ranges of measures that may be handled through fuzzy logic. A subset of the characteristic capabilities of the component systems is categorized and documented. Estimated costs, schedules and performance goals are established for the systems, interfaces, and SoS as a

whole. When the attribute models are combined in the SoS model, it is 'end-around' checked for consistency. At this point in the model development, adjustable parameters typically need to be adjusted in trial runs until the model is self-consistent but also in accord with stakeholders' goals. The completed model must be able to evaluate any proposed SoS instance within the meta-architecture for its KPA values and provide an overall assessment of the SoS. The relative worth of each attribute evaluation is described in the membership functions. The rules of the fuzzy inference system described how attribute values are combined for an overall SoS assessment. Visualizing the results of the characterization of the KPAs, the other inputs, the combination of systems, and the buildup of SoS capabilities from the component systems is the most useful part of the method for the SoS manager and stakeholders. Variations of all inputs, assumptions, rules, etc. may be examined to identify the most influential characteristics within the problem and to insure the formulation of the problem and solutions are proper and helpful. This approach can be used search for Pareto surfaces or other frontiers within the input and output spaces.

Figure 1.6 shows generalized steps for how to derive the set of attributes by which to evaluate the fitness of a selected arrangement of the systems and their interfaces to provide required capabilities to the SoS. The method determines the fitness of each architecture, or system + interface SoS arrangement, from the meta-architecture and domain dependent information. Attributes desirable in the completed SoS architecture are elicited from stakeholders through linguistic analysis of guided interviews with stakeholders. Having developed the attributes of interest, the possible ranges of evaluation in each attribute are separated into an agreeable number of gradations of

goodness or badness (defining the membership functions for fuzzy sets) with some overlap due to ambiguities in linguistic representation. The relative value of combinations of performances in each attribute is developed into fuzzy rules through a series of stakeholder hypothetical tradeoff exercises. The multiple objective optimization (MOO) problem of finding a good architecture over many dimensions may be solved by finding an architecture that maximizes the single fuzzy SoS fitness assessment. The method initially regards the independent variable to be the number of ones in a chromosome with randomly positioned ones and zeroes in it. The dependent variable is SoS fitness or overall quality. Exploring the architecture 'space' by evaluating a few hundred chromosomes with varying percentages of randomly placed ones provides insight into whether a solution within the constraints is possible. The rules and fuzzy membership function edges may need to be adjusted to find a set of tunable parameters that closes on itself (i.e., a set of parameters that produces solutions dependent on the architecture). When some good solutions are found to exist, a genetic algorithm approach is used to find a near-optimal arrangement from the meta-architecture. It is certainly possible to design a problem for which no acceptable solutions exist.

Combining all these steps into an organized method has not previously been applied to SoS. Because of the many simplifications in the method, it is not expected to directly provide final solutions but to give insight into behaviors of possible real solutions in response to changes in rules, definitions of capabilities, performance models, membership function shapes, environment, budgets, etc. that drive aspects of the development and evolution of SoS.

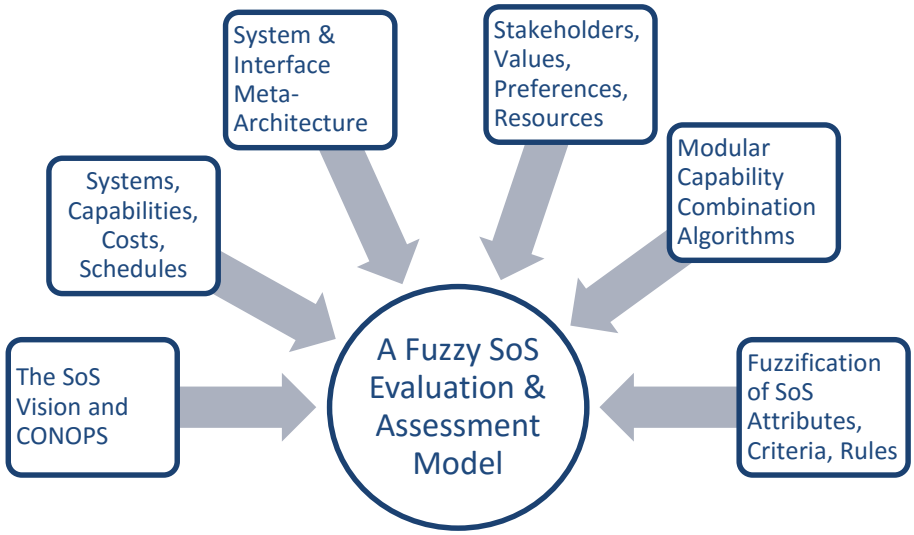


Figure 1.6. Overview of the contributions to the assessment model

1.9 SUMMARY OF FINDINGS

The proposed method was successfully demonstrated to find SoS architectures for a hypothetical intelligence, surveillance and reconnaissance (ISR) SoS in a Gulf War scenario, for an operation other than war (OOTW) scenario, for a search and rescue (SAR) scenario, for variations to a previously studied SoS model from MITRE called the Toy problem, and for an actual SoS of a large training program. Several variations of the method were used to look for Pareto surfaces, to conduct sensitivity analyses across a number of tunable parameters in the attribute models, and to examine the impact of changing parameters within the GA. Several useful visualization techniques were successfully implemented during the research. SERC Research Tasks 37, 44c and 109 *An Advanced Computational Approach to System of Systems Analysis & Architecting using Agent Based Behavioral Modeling*, sponsored this (and related) research into a wave model for acquisition of DoD acknowledged SoS (Dahmann, et al. 2011) (Dagli, et al. 2013). The evaluation and assessment algorithm portion of FILA-SoS was used on

architecture chromosomes developed by other members of the FILA-SoS team (using a non-gradient descent method instead of a GA), as well as by a Purdue/SERC team working on a counterfeit parts SoS. Other team members studied negotiation techniques to agree on a configuration under a range of environmental conditions between the systems and SoS manager. The part of the SERC research effort in this document describes the method to produce an SoS assessment method, and select a desirable architecture design for starting the negotiations to realize an SoS design to meet stakeholders needs.

1.10 CHAPTER ORGANIZATION

Chapter 1 introduced the importance and the need for methods to produce better models and to improve our understanding of issues involved in exploring the trade space when building an acknowledged SoS.

Chapter 2 is a literature review discussing what has been explored in the areas of fuzzy decision support tools, using fuzzy analysis to handle ambiguity in evaluation criteria, multi-criteria and multi-objective optimization, fuzzy genetic algorithms, and visualization in SoS architecting. There is no previous combined treatment of modular model building, coupled trade space visualization, meta architecture exploration and parameter tuning, and fuzzy genetic selection of architectures from an SoS meta-architecture.

Chapter 3 describes the model building method in detail, with worked out illustrations of the steps across several domains. The method of piecewise linear mapping the real attribute values to the fuzzy domain so that fuzzy models may be reused

is described. Use of the Matlab fuzzy inference system to describe membership function shapes and sizes is explained.

Chapter 4 discusses the results developed by using the method on the example SoS mentioned above. The FILA-SoS effort was transitioned to a large SoS problem proposed by Army Training Command in conjunction with system architects from MITRE. The model itself is proprietary (marked FOUO), but a sanitized version with system and capability names anonymized is included.

Chapter 5 contains some conclusions and a summary of the status of FILA-SoS with suggestions for future research.

The appendices contain an example of a more detailed ISR scenario, all the Matlab code for the attribute evaluations, the fuzzy inference system rules and membership functions, the input, output and display functions, a short explanation of DoDAF 2.0 model viewpoints, and additional illustrations of the input and output data for special cases.

2. LITERATURE REVIEW

2.1 SYSTEMS OF SYSTEMS (SoS)

Most of the work on understanding or developing SoS has 'approached from the side,' or looked at relatively narrow aspects of the problem as opposed to trying to understand SoS in their entirety. One of the problems with understanding SoS is that they frequently cross traditional domain boundaries. Either they address a broad new problem area that is not traditionally understood as being connected, or they develop because of changes in technology that allow for novel connections and unprecedented capabilities. Either way, analyzing this type of problem requires extensions to the old ways of thinking about problems. Simply describing the characteristics, boundaries, expectations, or governance of an SoS is difficult, being fraught with no commonly accepted terms for the new capability, little agreement on what constitutes success, nor even a good theory of SoS (Trans-Atlantic Research and Education Agenda in Systems of Systems (T-AREA-SOS) Project 2013). The acknowledged SoS that is the focus of this effort only exacerbates these problems because of the inherent limits in the responsibility, authority and accountability between the SoS manager and the system program offices that participate in the SoS formation (Director Systems and Software Engineering, OUSD (AT&L) 2008), (Pitsko and Verma 2012). The literature describing SoS engineering (SoSE) is growing in coverage, but it is still relatively sparse.

The differences between SoSE and systems engineering are discussed by Flanagan and Brouse (Flanagan and Brouse 2012), pointing out that different sorts of trade spaces open up in SoS. Some of the concepts about flexibility used here in section 3.6 trace to the discussion of options and limiting risk in DoD programs from Giachetti

(Giachetti 2012). Countering some of these difficulties in describing SoS architectures are the advances in describing complex systems with fuzzy sets (Gegov 2010).

There have been few attempts to describe architecting methods for a*cknowledged* SoS. One such approach is based on the federated architecture (FA) (Ahn 2012). FA is a pattern that describes the construction of a meta-architecture. This approach emphasizes features to allow interoperability and information sharing between component systems and the centralized controller. Another approach has been to model the interdependencies of systems and impacts of failures using Bayesian networks. An example is the outcomes of the Bayesian analysis with failure rates modeled as beta distributions providing a knowledge base for decision makers to control risk in development of an SoS with complex interdependencies (Han and DeLaurentis 2013). These examples still look at relatively narrow aspects of the SoS development problem.

Warfield introduced the concept of using binary matrices to describe system components' relationship with each other (Warfield 1973). That paper described how to construct reachability matrices using graphs representing directed interfaces, and a number of mathematical techniques to find compact regions in a general system representation of subsystems, but the last few paragraphs mention that this approach could also be used to show "objectives, events, activities, motors, generators, radars, etc.", or in this case, capabilities of elements of the SoS, or non-directed interfaces. There is undoubtedly more that can be done by extending the present research to directed graphs, however, the concept was borrowed for use here only to do the display of a much simpler approach to SoS architecting.

The SoS acquisition environment may be affected by external factors such as changes in the national priorities, changes in the SoS funding, or changes in threats to the nation, the business climate, or existing commercial arrangements. Clearly, foreseeable events should be accommodated through planning. The environmental changes spoken of herein are changes outside the framework of expectations. One traditional way to be ready for unexpected change is to have an abundance of spare capacity or capability, but that costs something. It costs something not only in resources devoted to carrying and maintaining the capacity beyond immediate need, but also in opportunities forgone. Introduction of this method may help allocate scarce resources better in the future cost constrained environment.

2.2 SoS ATTRIBUTES

Systems engineers call the areas of engineering design that require detailed knowledge and detailed analysis tools 'specialty engineering' areas (INCOSE 2011). These types of areas may also be called attributes of an SoS. Just as a measure of 'reliability' or 'availability' may require very detailed analyses at many levels within a system design, but result in a single overall number to characterize the design in that specialty area, the attributes of an SoS may require detailed analyses, but result in a single characterizing number. The attributes or specialty areas are sometimes be called '-ilities;' they are the subject of continuing, intense research, especially in the area of SoS. Large lists of the attributes, many with several definitions, are being catalogued and organized in several on-going efforts (Mekdeci, et al. 2014) (Ross 2014) (Ross, Rhodes and Hastings 2008). Just as that single number characterizing a system in a specialty area may have numerous conditions limiting its applicability, the attribute measures

characterizing the SoS will probably be valid over a limited range of scenarios. To understand the implications of a particular measure, one needs to know about all those conditions. Simply presenting that data in an intelligible format is a challenge. Finally, since the specialty engineering areas typically have well-known algorithms and procedures for evaluating combinations of subsystems that are easily extended to combinations of systems, this effort attempts to deal with more appropriately SoS specific attributes. These SoS attributes might be described as the ones which depend more heavily on the SoS systems and interfaces, which is detailed in the chromosome.

2.2.1 Attributes Commonly Found in the Literature. A key feature of the attributes of either systems or SoS is that they frequently pull in different directions. For example, improving speed may reduce range, both key attributes of overall technical performance. Improving reliability may increase cost, thereby reducing acquisition affordability, but possibly increasing operations and maintenance affordability. Numerous other candidate attributes of SoS exert pulls along different directions in the multi-dimensional design or architecture space. The selected architecture must satisfy the most unhappy stakeholder at least enough to avoid a veto. The stakeholders' concerns are represented in the attributes selected to grade the value of the proposed architectures. The models used to evaluate the attributes must be fully described and open to stakeholders so they can assure themselves the competition among architectures is fair. The weighting between attributes must be open and fair as well.

Pitsko and Verma (Pitsko and Verma 2012) describe four principles to make an SoS more adaptable. They spend a large part of their time describing what adaptable means to various stakeholders, that different stakeholders may continue to have slightly

different concepts of what adaptability means, that the definition is probably dynamic – changing over time, and that this ambiguity likely will apply to many other SoS attributes. Schreiner and Wirthlin discuss a partial failure to fully model a space detection SoS architecture, but learned a lot about how to improve the approach the next time they try it (Schreiner and Wirthlin 2012). The point is that people are not modeling according to a well-developed theory of SoS and then reporting on the success or failure: they are still attempting to define the theory.

There are numerous approaches in the literature attempting to describe useful attributes, as well as how to measure them, to help understand or predict the value of various architectural arrangements. These include evolvability and modularity almost as complementary attributes (Clune, Mouret and Lipson 2013), while Christian breaks evolvability into four components described as extensibility, adaptability, scalability and generality (Christian III 2004). Christian introduces the concept of complexity to overlay on these attributes because 'too simple' a system cannot evolve. Kinnunen reviews at least four definitions of complexity (Kinnunen 2006) before offering his analysis of one definition related to the object process methodology (OPM) of Dori. Mordecai and Dori extend that model to SoS specifically for interoperability (Mordecai and Dori 2013). Fry and DeLaurentis also discussed measuring netcentricity (interoperability within the SoS), noting also the difficulty of pushing the commonly used heuristics too far, because the Pareto front exists in multiple dimensions (Fry and DeLaurentis 2011), not merely two dimensions, as it is commonly depicted. Ricci et al. discuss designing for the evolvability of their SoS in a wave model and playing it out several cycles in the future, evaluating cost and performance (Ricci, et al. 2013). Because SoS are complex, there are

many ways to look at them, with no dominant theory yet. This is why this direction of research is interesting and worth pursuing (Acheson, et al. 2012).

Slightly different definitions for some of the SoS attributes were chosen for this work, especially for flexibility and robustness. Lafleur used flexibility in the operational context of changing a system after deployment (Lafleur 2012), which is to narrowly a system viewpoint to be used for the SoS. Robustness is used here in a different way than Deb and Gupta's classic notion of robustness (Deb and Gupta 2006), that is shifting the optimum point (defined as narrowly better performance), rather than accepting lower performance across a wider front – the path taken here. Singer used robustness in a different operational context (Singer 2006), that of losing a node in a network, rather more like losing a system or an interface from the SoS as described here. Gao et al. discussed a concept of robustness as the ability to withstand hacker attacks for 'networks of networks' with varying degrees of interconnectedness (Gao, et al. 2011). The concept of the flexibility attribute used here is more attuned to giving the SoS manager flexibility during development, when selecting systems to supply all the desired capabilities. This falls right in line with some recent discussions of resilience and sensitivity analyses, although they use the terms resiliency or robustness for it (Smartt and Ferreira 2012) (Yu, et al. 2011) (Jackson and Ferris 2013). The point is that there are many possible ways to describe the attributes of systems and even more ways for SoS, depending at a minimum on circumstances, organizations, and stakeholders' preferences. Many of these ways of thinking depend directly on the architecture of the system of interest. This dependency on interconnectedness fits into the framework of the architecture meta-model used here. If an attribute does not depend on the SoS architecture in any way, then it will not be

useful to help select between potential architectures. It is not necessary that a useful ranking algorithm be very accurate in its relationship to the measured attribute, only that it be reasonably well correlated to reality and nearly monotonic in its ranking. That is sufficient to be useful in this approach.

For purposes of this research effort, the following key attributes for a family of ISR SoS were defined by a group of subject matter experts (SMEs) during the SERC research task RT-44 (Dagli, et al. 2013):

- Performance: Generally, the sum of the performance in required capabilities of the individual component systems, with a small boost in performance due to increased coordination through interfaces. This is explained further in section 2.3 on netcentricity.
- Affordability: Roughly the additive inverse of the sum of the development and operation costs of the SoS. The performance factor above is occasionally applied in a different way to the affordability to change its shape as a function of the number of interfaces, but also to be somewhat related to superior performance.
- Developmental Flexibility: This is roughly the additive inverse of the number of sources that the SoS manager has for each required sub capability. If a required capability is available from only one component system, then the SoS manager's flexibility is very small; they must have the only system that can provide a required capability as part of the SoS. On the other hand, if each capability is available from multiple systems within the SoS, the manager has far more developmental flexibility.

- Robustness: This is the ability of the SoS to continue to provide performance when any individual participating system and all its interfaces is removed. Generally, having a very high performing system as part of an SoS is a good thing; however, if that system is ever absent, the performance of the SoS may be degraded substantially. Therefore, it may be useful to have the contributions of the individual system capabilities more widely dispersed, so that the loss of one system does not represent as great a percentage loss to the SoS (Pape and Dagli 2013).

2.2.2 Correlation of Attributes. There is a tendency for the quality attributes of systems (or SoS) to be correlated. A 'good' system or SoS by definition has many good attributes. This is not necessarily a natural condition; it takes considerable effort. Good architecting and design processes should result in this condition. Program managers with a good 'feel' for their problem area, whether systems or systems of systems (SoS) can often deliver good results. That 'feel' is difficult to duplicate or teach. This research is an effort to provide a way for a larger audience to be able to break down the problem to smaller, more understandable elements, and to build up the solution in a way that a wider group of stakeholders can understand and accept the discovered implications in the modeling.

If two attributes are highly positively correlated, then this is equivalent to counting one of them twice in the overall assessment. In an otherwise balanced design, counting one attribute twice is unfair to the other attributes, and may skew the design away from optimum. If their correlation is highly negative, then they tend to cancel each other out in an overall assessment, giving more weight to the remaining attributes than

they deserve. An ideal set of individual attributes would be strongly de-correlated, so they are measuring essentially different and independent aspects of the SoS. In mathematical terminology they would be orthogonal. It is part of the architect's art to select appropriate attributes. That is, to define them smartly, derive evaluation algorithms for them that depend on the architecture, and socialize all of this across the stakeholder community. This includes finding out what attributes the stakeholder community value, as well as discovering the relative strength of preferences among them. This act of discovery and elicitation can only occur through extensive discussion and focused probing. It also includes finding ways to calculate values for each selected attribute in a way that depends on the architecture. Not all attributes depend on the architecture, but many do. Only those attributes that depend on the architectural arrangements of the desired SoS should be included in the discussion of what SoS architecture is best. Attributes that do not depend on the architecture should be excluded from this portion of the planning.

The point of having different aspects in the SoS assessment is to achieve a balance among those different aspects. Furthermore, even if some variables are highly correlated, it need not imply that there are no differences between them. The fact of some modest correlation among the attributes does not mean that there are not still important differences, nor that there does not exist a 'sweet spot' that is the best compromise position among the conflicting desires of the stakeholders. This is also part of the fuzzy assessment process, where each of the proposed evaluation algorithms are explored across the range of values possible within the meta-architecture, to insure that they measure what is being sought. Additionally, appropriate membership function

names must be matched to the appropriate ranges of values, and the approach vetted among all stakeholders and subject matter experts. The modeling must tell a story, and appropriate, easy to assimilate, and to remember names for the parts help in this effort. Typically, this requires several model design iterations, with trial algorithms and adjustments to the boundaries between the quality attribute levels, or even trying different attributes, to get acceptable levels of fidelity. Equally important is to be able to explain the impact of having correlated attributes among the evaluation criteria of the SoS. Examples of how small changes in the architecture could change the evaluation by relatively large measures are relatively easy to find. An example of a very small change to the architecture could be removing one communications channel. That change of one bit in the chromosome would change many interfaces to infeasible from feasible, thereby changing the performance or the robustness very significantly. Showing these examples to stakeholders (and being able to explain them), are important elements of the socialization process to get prospective members of the SoS (or other stakeholders) to agree to support not only the SoS, but the modeling process. The member systems must support the values of the SoS analysis, because they typically give up something (hopefully small) within their original mission performance to be able to support the new SoS.

For example, improving one quality attribute, modifiability, might adversely affect another quality attribute, performance through increased latency, then the range of acceptable values where modifiability induced latency does *not* adversely affect performance must be defined, along with how a layered architecture, which might impact modifiability, also impacts latency. Other architecture properties could also impact

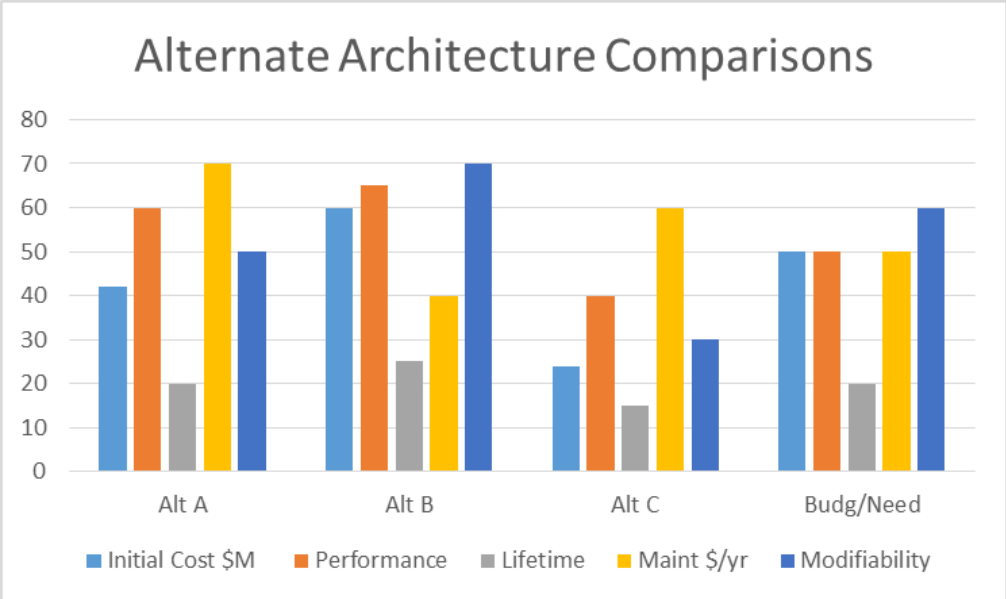
latency as well, such as volume of data being exchanged or capacity of the communications link. The proposed method addresses this in the step for modeling the attributes as a function of any selected architecture (within the meta-architecture framework). The correlation might be negative, but acceptable values of both (simultaneously with other attributes) must be achieved to have a viable SoS architecture. The SoS architecture description and domain dependent system data should show how the different aspects of the design (attributes) impact each other, are self-consistent, and most importantly, are simultaneously achievable.

There are many ways to illustrate the impact of attribute values on the SoS quality. The data must be conveyed to decision makers, whether architects, designers, managers, or key stakeholders. An impediment to correctly ranking the overall architecture based on several attributes is shown in Figure 2.1 and Figure 2.2 when some attribute values are better smaller. Lists of values, stacked bar charts, or a Kiviat chart, such as those shown here for example. Both these examples show some attributes (e.g., cost) that are better when smaller, and others (performance) that are better when larger. Figure 2.2 shows relative architecture comparisons on a scale of 10 as the desired value. An important part of the architects' skill is to find a way to show all attributes better in the same direction. This is shown in Figure 2.3, where costs have been transformed into affordability; one can much more easily determine that Alternate B exceeds desires in all areas except affordability, and the very affordable Alternate C is less than desired in all other attributes. None of these displays clearly identifies a 'best' alternative. That is still very much a subjective decision, even in Figure 2.3. Neither do they indicate the sensitivity of an attribute between the alternatives. For example, perhaps one could trade

some performance or modifiability for affordability in Alternate B. When the sample alternatives being compared are SoS architectures, much information is necessarily hidden in these views, yet any of these views are relatively difficult for decision makers to comprehend. This is why the proposed method includes significant effort to discover, understand, document, and socialize the meaning of terms and evaluation algorithms used throughout the model. Gentry Lee, chief engineer of the Jet Propulsion Laboratory, says "The systems engineer must know the partial of everything with respect to everything else" (NASA 2007). This includes quality attributes, technical data, performance predictions, implications of proposed changes, costs, schedule and customer valuations. In areas of confusion, uncertainty or disagreement, the sensitivities can more easily be explored to find the 'best' (or at least a close to best) architecture because the method creates and records the open algorithms and data to evaluate all the attributes for any configuration within the meta-architecture.

If an acceptable and achievable SoS is not found, then analysis should help one decide how close or far any particular instance is from acceptability or achievability. The analysis should also give indications of which attributes must be improved to be acceptable, as well as what changes to the architecture could move it in the 'right' direction. Kiviat charts (or in Excel, radar charts) allow one to see several project measures simultaneously; but even with a well-designed chart, it may be difficult to decide which is better between two (or more) alternatives with this visualization method.

Attribute	Alt A	Alt B	Alt C	Budg/Need
Initial Cost $M	42	60	24	50
Performance	60	65	40	50
Lifetime	20	25	15	20
Maint $/yr	70	40	60	50
Modifiability	50	70	30	60



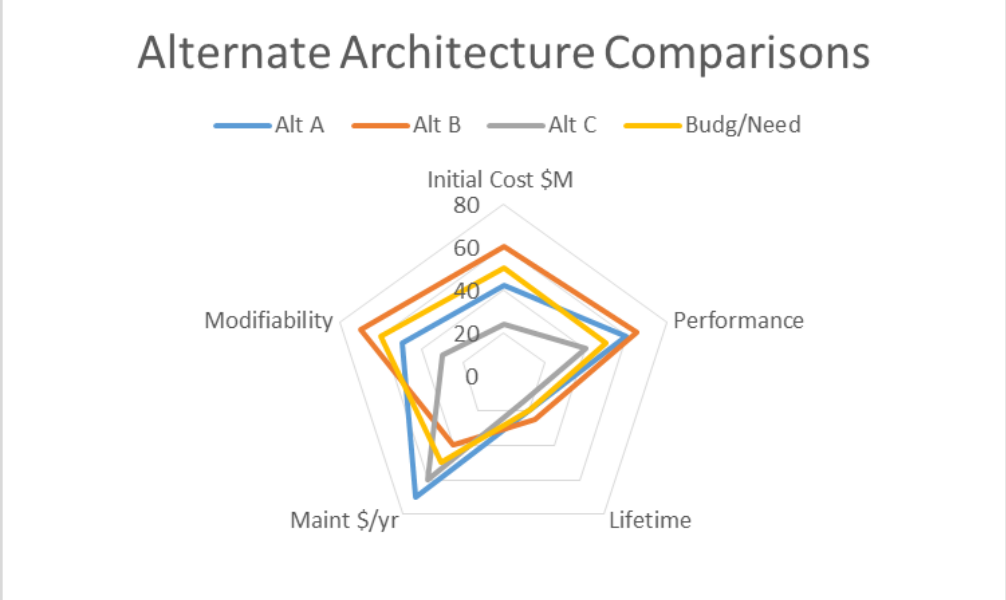
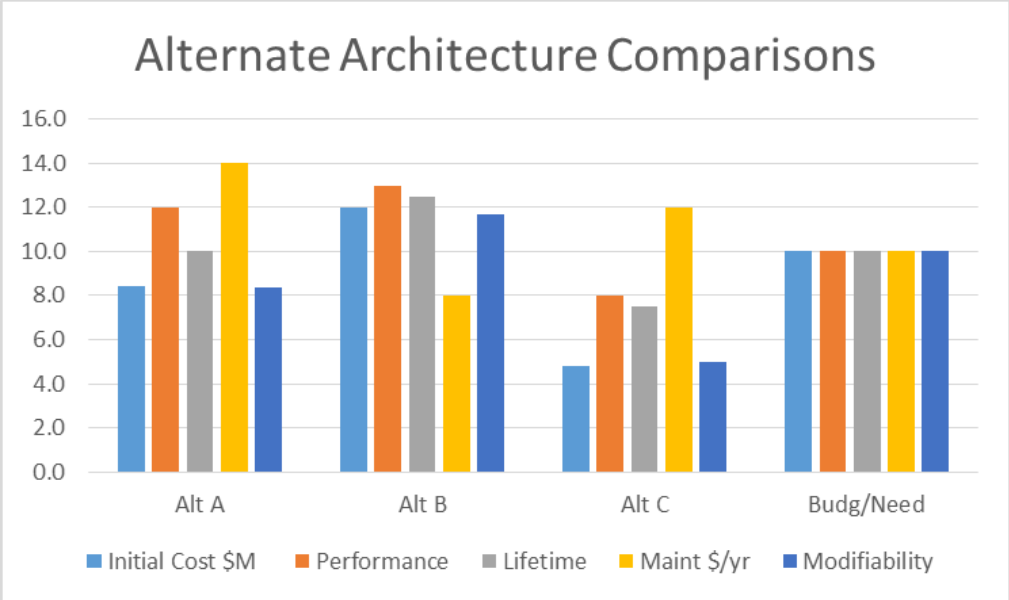


Figure 2.1. Absolute architecture comparison illustration

Attribute	Alt A	Alt B	Alt C	Budg/Need
Initial Cost $M	8.4	12.0	4.8	10.0
Performance	12.0	13.0	8.0	10.0
Lifetime	10.0	12.5	7.5	10.0
Maint $/yr	14.0	8.0	12.0	10.0
Modifiability	8.3	11.7	5.0	10.0



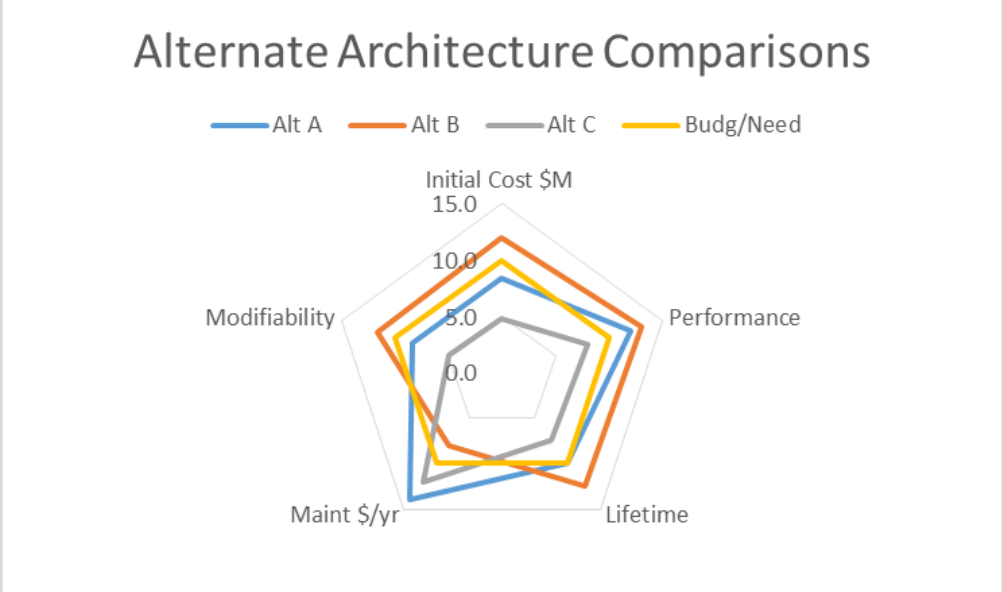


Figure 2.2. Relative architecture comparison

Attribute	Alt A	Alt B	Alt C	Budg/Need
Affordability	11.6	8.0	15.2	10.0
Performance	12.0	13.0	8.0	10.0
Lifetime	10.0	12.5	7.5	10.0
Maint Afford	6.0	12.0	8.0	10.0
Modifiability	8.3	11.7	5.0	10.0

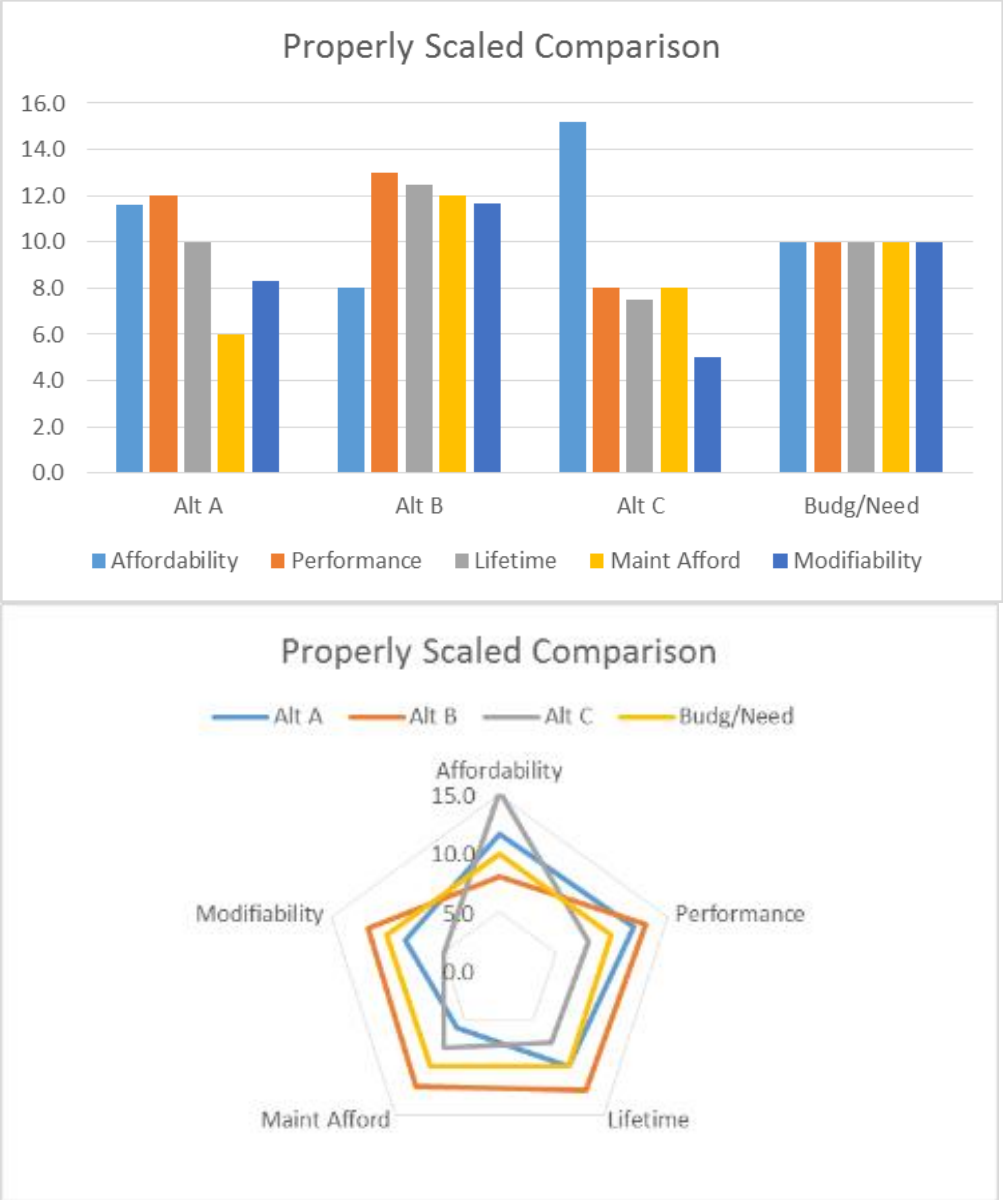


Figure 2.3. Properly scaled architecture comparison; still not conclusive

The practice of creating a series of simplified functional relationships between the meta-architecture presence or absence of the systems and interfaces, represented in the chromosome for each of the key attributes is the key to the proposed assessment method. It can demonstrate to the stakeholders the implications of systems' choices to participate and how many interfaces to pursue. The process of randomly filling in the meta-architecture with ones and plotting the resultant attribute values allows everyone (from analysts to program managers, to funding stakeholders) to see values or costs of participation. By sorting the plots by the number of ones in the SoS meta-architecture, the process illustrates how changes in each part of the model contribute to the overall SoS quality: systems count, interfaces, definitions of capabilities, how individual capabilities are joined to build the SoS capability and performance, as well as attributes, membership functions, and rules for combining the attributes. Furthermore, by exposing the inner workings of the component models to everyone, the strength of the architecture model construct is far stronger than the historical practice of PowerPoint engineering through even the formal architecture tradeoff analysis method (ATAM) practice of having outside SMEs comment on risky parts of the architecture. At least ATAM provides a series of checklists for items to review about the architecture.

Figure 2.4 shows the impact of changing the number of ones in a population of 5000 chromosomes for the ISR case described in section 4.1.1 with 22 systems. There are clear trends in many of the quality attributes, but also much variation within them, even from one chromosome to the next in the series (ordered by the total number of ones in the chromosome). The same number of ones could be distributed differently between systems and interfaces, as well as between different sets of systems, or different

arrangements of interfaces for the same set of systems. Only the total number of ones in the chromosome is tracked as the independent variable in this portion of the analysis.

Table 2.1 shows the correlation coefficients between all the variables plotted in Figure 2.4, which is a relatively thorough exploration of the ISR SoS meta-architecture space. It seems remarkable that the highest correlation between an attribute (performance, labeled 'perf' in column 4 of the table) and overall SoS quality (labeled 'sos' in row 3 of the table) is as weakly correlated as it is (0.1459). The only relatively high cross-correlation between any of the quality attributes is between flexibility ('flex' in row 5) and affordability ('afford' in the last column), at about $r = 0.8$; this doesn't even qualify as strongly correlated ($r^2 > 0.8$). Furthermore, affordability and flexibility are not closely linked in the way they are calculated, so even the slightly more than weak correlation that is seen here is questionable.

The rule-based fuzzy inference system approach provides a mathematically rigorous method to make architecture comparisons. This hinges on the individual attribute evaluation algorithms being well defined, and on the fuzzy inference system (FIS) for combining the attribute measures to the overall SoS assessment. The assessment is a single, composite, characteristic SoS value from the multiple attributes. This transforms the multi-objective problem into a single valued SoS function that can be optimized. The attributes and the FIS are developed through facilitated individual stakeholder discussions, including the SoS manager, then are vetted and socialized across the stakeholder community. The method is ideally suited to sorting through many candidate SoS instantiations from a meta-architecture of potential SoS designs in a way that is traceable and understandable to all the stakeholders.

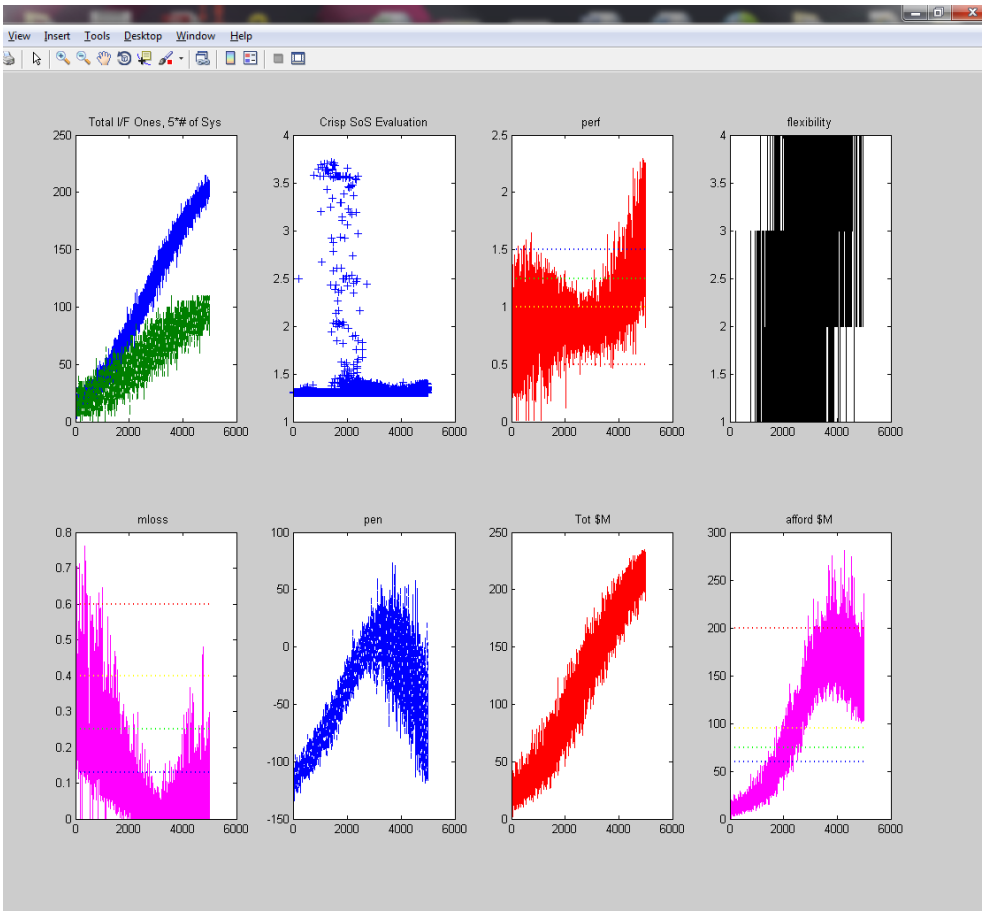


Figure 2.4. Exploring the meta-architecture space with varying participation ratios

Table 2.1. Correlation coefficients between attributes in Figure 2.4 are shaded

	pop #	i/f	sos	perf	flex	maxloss	sys	penalty	cost	afford
pop #	1	0.9915	-0.0176	0.3663	0.8352	-0.5671	0.9318	0.6565	0.9749	0.8955
i/f	0.9915	1	-0.0283	0.3515	0.8471	-0.5555	0.9306	0.6519	0.9796	0.9104
sos	-0.0176	-0.0283	1	0.1459	0.0779	-0.0049	0.0451	-0.0372	0.0133	-0.0337
perf	0.3663	0.3515	0.1459	1	0.3625	0.0395	0.5595	-0.2584	0.4491	0.0734
flex	0.8352	0.8471	0.0779	0.3625	1	-0.4544	0.8664	0.5124	0.8725	0.8023
maxloss	-0.5671	-0.5555	-0.0049	0.0395	-0.4544	1	-0.5368	-0.6598	-0.5235	-0.593
sys	0.9318	0.9306	0.0451	0.5595	0.8664	-0.5368	1	0.5385	0.9683	0.8261
penalty	0.6565	0.6519	-0.0372	-0.2584	0.5124	-0.6598	0.5385	1	0.6051	0.8439
cost	0.9749	0.9796	0.0133	0.4491	0.8725	-0.5235	0.9683	0.6051	1	0.8949
afford	0.8955	0.9104	-0.0337	0.0734	0.8023	-0.593	0.8261	0.8439	0.8949	1

2.3 NETCENTRICITY OF SoS

The acknowledged SoS being considered herein are inherently netcentric. Information is the primary resource exchanged across an interface. This approach heavily weights the presence of interfaces to promote interoperability and collaboration in addition to simply summing the systems' individual capabilities. The purpose of the concept of netcentricity is to achieve increases in performance greater than linear in the number of systems (Alberts, Garstka and Stein 1999). In other words, the SoS exploits the potential synergy of the combined systems to achieve greater performance through their working in a coordinated way. This coordination comes through exchanging information on sensor data, intentions, positions, etc., between systems, so that previously independent systems can coordinate their activities to be more effective (Alberts 2011) (Cloutier, Dimario and Polzer 2009). This concept may flow into other types of acknowledged SoS such as supply chains, intermodal transportation systems, health care, etc. In general, more interconnections mean more powerful synergies in the SoS (not taking this argument to the point of clogging the network/roads/etc. with too much traffic –that is a different issue than having the pathway merely exist between the nodes). One way to handle this improvement in performance from interconnections is to have very detailed models of every system and interface. Another way is to treat the interfaces generically and assume each one helps the overall SoS performance by a tiny fraction. If one does not count the interfaces at all, the SoS performance, $\boldsymbol{P_{SoS}}$, is simply the sum of the individual system performances, $\boldsymbol{\Sigma P_{Systems}}$. Allowing a slight improvement, ϵ, in the performance of the SoS from each interface in the meta-architecture is quite simple, as shown in equation 1. It is not a very accurate model, but it

makes some intuitive sense and shows a general trend of increased performance through improved interoperability.

$$\boldsymbol{P_{SoS}} = (\sum \boldsymbol{P_{Systems}}) * \ (\mathbf{1} + \ \boldsymbol{\epsilon})^{(\sum \boldsymbol{Interfaces})} \quad (1)$$

Whatever performance the systems can bring individually, the performance of the SoS is increased by a small amount when multiple systems act cooperatively through interfaces. Epsilon is a small fraction, approximately 0.1% to 1% of increase in the simple sum of the systems capabilities before accounting for interoperability. The sum of the interfaces can be scaled by a constant if the number of systems grows large. Adjusting both the scaling factor and epsilon allows fine control of the total netcentric improvement effect. This does not seem unreasonable. The addition of one interface does not change an overall SoS performance very much. However, when larger numbers of component systems are considered, potential interfaces increase proportionally to the square of the number of systems. Therefore, the impact of large numbers of interfaces within the SoS can be significant. This is the basic premise of the netcentric warfare movement, even though it ignores several criticisms and problems (Alberts, Garstka and Stein 1999). It has the advantage for this research of providing a performance difference in the model that depends significantly on the meta-architecture. If better models of the impact of adding systems and interfaces are developed or available, they can be substituted into this very simplified, generalized, but also moderately nonlinear SoS performance attribute model.

2.3.1 Achievable Interfaces Through Communication Systems. To prevent this SoS analysis method from being a simple counting exercise, a further complication is introduced through a new concept of 'achievable interface.' Here, achievability means

requiring a common communication link to enable the interface between systems to be achieved. In this way, the concept of a netcentric performance improvement is modified by only rewarding the use of achievable interfaces. Attempted use of unachievable interfaces, , that is, by having a '1' bit in an interface position in the architecture that is not supported by the appropriate communications link and interfaces, is now penalized. The reward or penalty depends both on the intention of having an interface (a '1' in the meta-architecture between systems), but also on the existence of a common communication link through which an information exchange takes place. The possible communication links are enumerated both as component systems within the SoS and as capabilities within the systems.

The meta-architecture is filled with random bits during the genetic algorithm approach to exploring the SoS architecture space, so there may be 'interfaces' that are not supported by communication links – therefore they are unachievable. Within a real SoS, a SPO may have spent resources to develop an interface. They might install equipment, antennas, make software changes, test the new configuration, etc. If the system on the other end of that interface did not also install the interface, there is not a real interface there. If both systems do the development work for the interface but the communications system is not available during operation, due to jamming, not having a relay system, or lack of cryptographic compatibility on that day, then again – there is no real interface, i.e., no information exchange is achievable over that interface. The communication system might be down for maintenance, filled with higher priority messages, compromised by hackers, a system might lack the encryption keys they need to use it, or any number of other problems prevent the use of an interface. In all these cases of

unachievable interfaces, any equipment a system carries to make this link possible also carries a penalty to normal performance. A size, weight, power, cooling, fuel, payload, range, throughput, or memory penalty is paid to carry the unusable interface. Therefore, having the performance reward or penalty increment depend on the achievability of the interface seems quite reasonable.

In netcentric SoS, the interfaces are normally through communication links. The communication links are a special type of system within the meta-architecture. Since the location and number of ones in the chromosome are the independent variable in the GA approach, a pair of systems may say they have an interface, when there is no possibility of achieving it, because there is no common communication system between them. Therefore, the 'achievable' interface is one where the two systems must interface through a communication system in common. In order to get credit for an interface as a performance improver, both systems must be present, their interface bit must be a one, and in addition, both systems must have an interface with a communications system in common, as shown in Table 3.1 and in Figure 2.5.

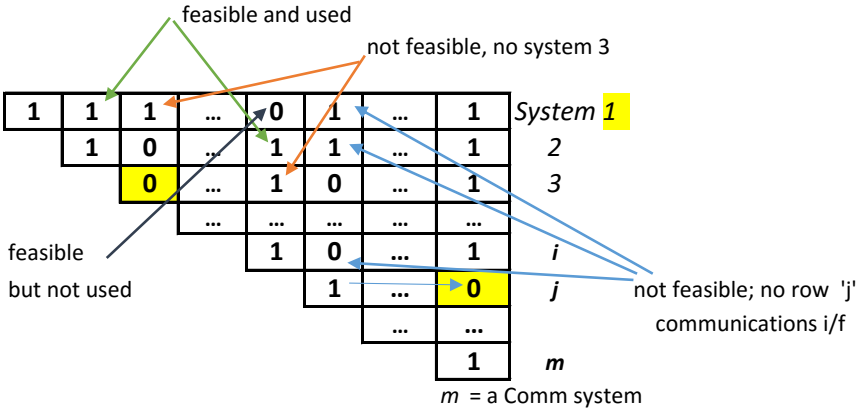


Figure 2.5. 'Achievable interface' has a communication system path in common

2.3.2 Special Treatment for 'Linking' Systems. One of the primary ways that systems interface with each other is through communications links, this depends on the domain of the SoS. A transportation SoS may link through switching yards or intermodal transshipment points; a chemical refinery SoS might interface through manifolds and valves. These elements could be considered systems in their own right, or nodes in the graph of a network. Most of the systems considered herein accomplish their interfaces by exchanging information. The acknowledged SoS is normally created by joining independent, existing, mobile systems that achieve an interface primarily by exchanging information. Therefore, they do so through communications links. Most systems can use multiple communication systems, as well. The simple, general meta-architecture model discussed so far is modified by adding a rule for placement of the communications systems, and another rule for the interfaces between the remaining systems as follows:

1. Gather the communication systems to the bottom in the list of systems
2. Insist that a 'claimed' interface between system i and system j (X_{ij} =1 in Figure 1.4) be supported by mutual interfaces to a common communications link (system k) from both system i and system j (∃ k, X_{kk}=1, X_{ik}=1, X_{jk}=1). If so supported, X_{ij} =1 is called a *achievable* interface; if not so supported, it is *unachievable*.

Rule 2 allows postulating an increase in the measure of an attribute for using achievable interfaces in the netcentric SoS, and penalties for unachievable interfaces. This conception of the interface matrix separates even further from the adjacency matrix paradigm, because it is not a simple graph with the addition of the second rule. It might

be possible to break the matrix into separate simple graphs, but this is not a graph-theoretic discussion, so there will be no further speculation in that direction.

Given the meta-architecture as described in section 1.6, with the above additional rules on interfacing through a communication system, the netcentric SoS as defined here is a reasonable model of an acknowledged SoS. Using the concept of achievability described above, a reward or penalty function may be defined to recognize the impact of netcentricity (or implemented interoperability) on performance or other attributes of the SoS. This allows for an evaluation of useful SoS attributes directly from the meta-architecture. This approach is not previously found directly in the literature.

2.3.3 Improved Netcentric Performance Equation. Interfaces can have either positive or negative impact on $\mathbf{P_{SoS}}$, due to the concept of achievability. In addition, the ratio of penalty to reward for each type of interface is one of the adjustable parameters in the performance attribute model. But, performance is not the only factor in suitability of an SoS. Adding new interfaces always costs something, so they detract from affordability whether feasible or infeasible. The impact on overall SoS assessment, by the addition or subtraction of an interface from the chromosome (see section 2.6) is difficult to predict for the other attributes, because an individual interface is not strictly nor straightforwardly linked to the other attributes in a simple way that can be interpolated.

A systems engineer, designer, or architect can use this information is to guide their exploration of the trade space in the SoS meta-architecture. They may use it to challenge assumptions, policies, or any of the component pieces of data in the model. In the examples it is used to look at correlations between selection of individual systems,

changes in input data, rules of how long operations costs count, membership function boundaries, or changes in the algorithms for evaluating each attribute, in the overall SoS assessments.

Even with no interfaces, adding individual system performances increases the performance of the SoS, $\mathbf{P_{SoS}}$, linearly. However, since the performance of the SoS is affected by the number of interfaces, it is possible that performance may be improved even more through interfacing a fixed number of systems than by adding more systems alone. The actual performance algorithm of a new SoS may be as simple or as complicated as required; there is no requirement that it take this form. The purpose of using this particular performance equation is to have something complicated enough to fully exercise the modeling method. It is representative of potential SoS performance measures. It is non-linear, which was a self-imposed goal to show the method works for non-linear combinations of systems and interfaces. The equation for the demonstration performance of the SoS should more properly be written as follows:

$$\boldsymbol{P_{SoS}} = \left(\sum_{i}^{m} \boldsymbol{P_{System(i)}}\right) * (\mathbf{1} - \boldsymbol{\epsilon})^{(\boldsymbol{Penup} * \sum \boldsymbol{Unachievable\ Interfaces} - \boldsymbol{Pendn} * \sum \boldsymbol{Achievable\ Interfaces})} \quad (2)$$

Where ***Penup*** is the scale factor for increasing the penalty for using an unachievable interface, ***Pendn*** is the scale factor for decreasing the penalty for an achievable interface (or increasing the reward for a good interface). The sign of the netcentric boost, $\boldsymbol{\epsilon}$, was reversed from equation 1 to fit the penalty/reward paradigm

instead of a pure reward paradigm of equation 1. The sums of the achievable and unachievable interfaces are simply counted from within the chromosome.

After the initial introduction of the netcentric concept, the explanation should have continued on to say the exponent in the $\boldsymbol{P_{SoS}}$ factor consists of the sum of the achievable interfaces, minus the sum of the unachievable interfaces in the chromosome as shown above. The additional tunable parameters ***Penup*** and ***Pendn*** allow the improvement ratio of feasible to infeasible performance to be altered depending on the scenario. The factor can now go negative in the exponent, causing a loss in overall performance of the SoS when infeasible interfaces outnumber feasible interfaces (or not, if the ***Penup*** to ***Pendn*** ratio is not one-to-one).

The point of the exercise was to have a representative function for performance that depended in a reasonable way on the degree to which the architecture was interconnected. There is a considerable body of study in the Command and Control Research Program (CCRP) Network Centric Warfare (NCW) series (Alberts and Hayes 2005) (Alberts 2011) on how connecting an SoS can allow it to self-organize and improve its performance well beyond the simple sum of individual system performances. There is no agreement on what the improvement factor should be in general, because that would be highly system and scenario dependent. A factor of two to three improvement in effectiveness, however, seems eminently reasonable in the generic case.

In a real world validation problem using the Army training system from MITRE (see section 4.1.5), there were no key performance attributes (KPAs) that used the netcentric form of performance dependence, so it most emphatically is not required for the method.

2.3.4 Why Not Graph Theory. The FILA-SoS meta-model chromosome looks similar to the upper half of an adjacency matrix. An adjacency matrix of the graph **G**, written **A(G)**, is the n-by-n matrix in which entry $\mathbf{a_{ij}}$ is the number of edges in **G** with endpoints $\{\boldsymbol{v_i}, \boldsymbol{v_j}\}$, where **G** is a loopless graph with vertex set $\mathbf{V(G)} = \{\boldsymbol{v_1}, \ldots, \boldsymbol{v_n}\}$ and edge set $\mathbf{E(G)} = \{\boldsymbol{e_1}, \ldots, \boldsymbol{e_n}\}$ (West 2000). An adjacency matrix is symmetric, so occasionally notation is used that ignores the lower half just as the upper triangular form of the chromosome does.

In the FILA-SoS approach, the nodes would be the potential systems of the SoS (along the diagonal of the matrix) and the edges would be the interfaces between systems (in the upper triangular portion of the matrix). A 'one' in an interface position makes the two systems potential graph 'neighbors,' but there is a twist to that simple interpretation required by the condition that potential systems may choose not to participate in the SoS. The interfaces alone could be represented by an adjacency matrix, which either ignored or removed the diagonal (and assumed the diagonal was filled with ones). Since the diagonal represents all potential systems and some may not participate, it is important to have both the ones and zeroes in the diagonal. The adjacency matrix represents edges (connections) between existing nodes (systems). That interpretation assumes all the nodes in the graph exist. By introducing the concept that some of the potential nodes may not exist in the SoS, the straightforward interpretation of the upper triangular matrix (above the diagonal) as being the upper half of the adjacency matrix, is lost.

If one were to keep the simple graph interpretation of the interface as being an edge of the graph, and forced the system (node) interconnections (edges or interfaces) to be through the special intermediate nodes of communication systems, one could interpret

an interface as a collection of two edges with the special communications system node between them. This would still not account for missing nodes represented by the zeroes along the diagonal. Leaving the unachievable nodes in the graph allows the genetic algorithm (GA) to be implemented very easily. The introduction of the 'interface through a communications system' concept also complicates the otherwise simple interpretation of the matrix as described next.

At the end of the list of component systems is a special class of systems that also count as capabilities – these are the communications systems, or 'linking' systems. Other systems may have the communication links as capabilities or sub-systems, but also require an interface with the communication link system to count in my formulation of achievable interfaces. In today's environment, many communications links are accomplished through networks as opposed to being point to point. Two systems could even be connected through the same waveform, on the same network, but not within the same community of interest or secure subnet; therefore, they would still not be able to communicate. Therefore, to complicate the otherwise simple notion of a bipartite graph of the interfaces, a requirement is introduced that systems that claim an interface must both also have a valid, common, communications system interface as well. It might be possible to squeeze the situation back into a graph theory interpretation by rearranging portions of the meta-architecture matrix into two different graphs, one having only non-communication systems and the other containing the communications systems, where the communications system graph was not bi-partite. This seemed to be an overly complicated approach, and was abandoned.

The concept of the achievability of the interface accounts for situation of a 'down' communication system. Two systems may claim to be connected (by having a 1 in the architecture chromosome representation at the correct place for the interface), and be prepared to use any information shared with each other. Yet the systems will still not be able to do so unless they also have a common, working communications link. While an adjacency matrix would show which systems are connected to each other in the same way that the FILA-SoS meta-architecture shows the first order interfaces, it does not aid this second check through the communications link interconnections for each interface. The last column in Figure 2.5 shows the interfaces with a communications system, system *m* in the ***m***th row (and column). In order to decide if a 1 in an interface position of another row system is achievable, one must proceed both right to the communications system interface with that system, as well as down to the other system on the diagonal, then right to the communications interface with the second system. If both systems have a 1 for their interface with the communications system, then the original interface is 'achievable.' Having an achievable interface is good. Having an unachievable interface in the chromosome is bad. Essentially, it means one or both of the systems prepared for an interface, possibly modifying software or displays, or adding a radio or antenna, but still cannot exchange data with the intended partner for their trouble. It is a waste of resources, in both development and operation. Figure 2.6 shows the achievable and unachievable interfaces in an example chromosome, and whether they are used (represented by a 1) or not used (by a 0) through the color coding.

Because this achievability factor is added to the matrix, it changes from a straightforward 1st order interface representation between systems to something more

complicated, and at least is no longer simply 1st order for some of the interfaces. This breaks the simple connection to an adjacency matrix representation. This is not to say that an adjacency matrix representation couldn't be used, only that it seems easier not to do the problem in graph theory notation, instead using the matrices as place holders for similar and closely aligned, but different types of information.

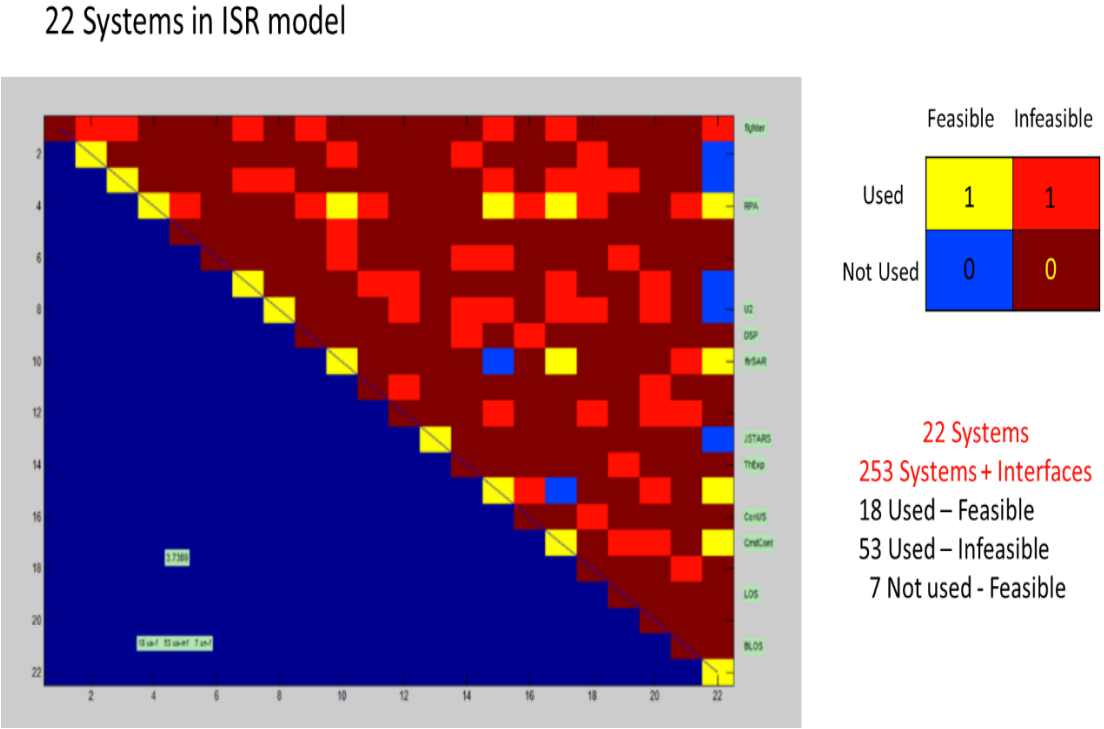


Figure 2.6. Achievable/unachievable, and used/unused interfaces

The systems vs. capabilities matrix of the required input domain data is indeed an incidence matrix, showing which systems have which capabilities, examples shown in Figure 2.7 and Figure 2.8.

CapName	Cap-Sys1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
EO/IR	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
SAR	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
Exploit	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
C2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
Comm	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 2.7. Incidence matrix for systems vs. capabilities for the ISR SoS

CapName	Cap-Sys1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
IR – range 3 nm	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Night Vision – range 3 nm	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Visual – range 3 nm	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
Maritime Radar – range 30 nm	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
RF Direction Finding – range 70	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
Deliver Medical Aid (Deliver Pa	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
Remove survivor(s) to Emerger	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
Speed 300 mph	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Speed 15 mph	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
Communication	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 2.8. Incidence matrix for systems vs. capabilities for the SAR SoS

The advantage of having the architecture composition problem formulated in graph theory, with the matrix representations actually being adjacency matrices would be that a large body of graph theory mathematics already exists for manipulating, understanding, and applying the matrices in certain classes of problems that would no doubt be useful in solving SoS architecting problems.

2.3.5 Why this is Not a Simple Assignment Problem. One could interpret capabilities as tasks and systems as doers of tasks in the classical assignment problem formulation to approach this problem. In the entire FILA-SoS architecture approach, the systems get to negotiate adjustments to how much they will participate, given a decision to participate, and also have freedom to withdraw completely. This is very different than the classical assignment problem. The assignment problem seems to be more oriented toward centrally controlled systems than the loose confederation of the acknowledged SoS. However, the fuzzy GA architecture selection part only recognizes the freedom to choose not to participate by the presence of zeroes in the chromosome. It might be possible to formulate this part of the problem as an assignment problem but there did not

seem to be away to assign the constraints in the assignment problem format with potentially missing systems, or non-linear performance based on the participation. There are ways to formulate a multi-objective assignment problem by combining the multiple objectives in exactly the same way as the fuzzy assessor (De and Yadav 2011), but that paper still assumed a one-to-one connection of tasks and performers. Furthermore, the way the capabilities of individual systems are joined together in the SoS is not as straightforward as in a typical assignment problem. It is not simply assigning systems to tasks (or capabilities) on a one-to-one basis. The way that capabilities are joined in an SoS could be quite nonlinear, and vary depending on which systems bring which of their possible multiple capabilities together. A method could not be determined for how to assign dummy tasks in the assignment problem formulation for missing systems or interfaces. It may be possible, but did not seem to be a general way to do it. The GA approach was much easier to formulate on the meta-architecture.

At any rate, the purpose of the proposed method is not to provide a final, truly optimum design for the new SoS. The purpose is to explore the impacts of policy changes, different environment situations, changing choices of acceptable levels of the key performance attributes, choosing entirely different KPAs, or valuing KPAs differently within the fuzzy rules, on the selection (through genetic optimization methods) of 'good' architectures. The design analyses permissible with this very simplified model with many adjustable parameters is limited to evaluating instances of the meta-architecture. Given a set of input data, the meta-architecture is limited to a binary presence or absence of the possible systems, and first order direct interfaces between each of those systems. The number of possible architectures (or chromosomes)

with this formulation is $\mathbf{2^{m(m+1)/2}}$. Even if numerous heuristics are employed to help select a 'good' architecture, it is very difficult to do this for multiple KPAs simultaneously. For example, it's not hard to write rules (heuristics) to select high-performing systems with low costs (i.e., good affordability); however, these choices perform poorly in the attributes of robustness (still good performance with a missing system) or flexibility (multiple systems can provide each capability). The method was developed to be as heuristic free as possible, because it is not understood what the right solution to this problem will be yet, and therefore one cannot know which heuristics will be the useful or appropriate ones.

Heuristics clearly can help find solutions more quickly, and the discovery of heuristics is important to finding better and/or faster solutions to many types of problems (Maier and Rechtin 2009). However, by definition, the reason a heuristic works is not strictly known (Blanchard and Fabrycky 2010). Heuristics may bias the discovered solution by discarding possibilities in unknown ways. Even though many heuristics are known to be biased, they are used both intentionally and unconsciously (Taleb 2004). There are no guarantees that any particular heuristic will continue to be useful (as it has been in the past) on a new type of problem. Heuristics are common sense derivations from experience in solving similar problems, but if the reason they worked were fully understood, they would be part of the formal solution method and not classed as an heuristic. The methods worked out here attempt to limit heuristics because the nature of a 'good' SoS solution is not yet understood well enough to trust any heuristics. The example problems are not large enough to require extensive use of heuristics to reach a reasonable solution in quite reasonable times, either, which is a standard reason for

relying on an heuristic: to narrow the search space and reduce the time to compute a solution (Blanchard and Fabrycky 2010). In the proposed method, heuristics might be unintentionally embedded in the attribute definitions, evaluation algorithms, membership function shapes, and fuzzy rules, but every attempt was made to avoid heuristics.

2.4 FUZZY LOGIC.

2.4.1 Just Enough Fuzzy Logic. The fuzzy logic systems used in this research are quite basic. Simple Type I fuzzy sets are used throughout (Zadeh 1975) (Mendel 2013) (Dauby 2011). The intent was to discover and demonstrate the usefulness of a fuzzy logic approach in reasoning about finding 'good' SoS instances from a simplified, binary meta-architecture. For this reason, the simplest possible triangular membership functions were used at the beginning (Singh 2011). As the research progressed, it became clear that trapezoidal membership functions were equally simple to use within the Matlab Fuzzy Toolbox and were also more nearly a match for SoS acquisition and design reality. Matlab '.fis' files that detail the fuzzy membership and rule bases of each file are shown in Appendix C.

2.4.2 Impact of Recent Advances In Fuzzy Logic. The approach used in FILA-SoS was limited in many ways. The first way was that the meta-architecture included only binary (i.e., fully in or fully out) participation by the systems, and secondly, only first order, non-directed interfaces. Only Type I fuzzy systems were used, with limited ranges of overlap of the degree of membership in adjacent Gaussian rounded trapezoidal membership functions. The rule sets in the fuzzy inference system implementations were kept to a minimum, while allowing enough non-linearity to show that linearity was not necessary, but no more. The purpose was to demonstrate the

validity of the fuzzy assessor concept working with the genetic algorithm over the simplified meta-architecture. The examples produced 'good' architectures over a wide range of input data values.

More recent concepts being developed within fuzzy logic could certainly be used to make better models. For example, if there were still discrepancies in the common understanding of the meaning of the attributes and various membership functions among the stakeholders, then it might be more appropriate to use Type II membership functions. This would allow the edges and/or shape of the membership functions to have an uncertainty or probabilistic character. This is one way of handling that type of uncertainty. Golkar suggests a number of ways to elicit information from SMEs in cases of large ambiguity (Golkar and Crawley 2014) which could mesh with either Type II fuzzy systems or interval valued fuzzy systems. Using interval fuzzy sets would allow the uncertainties of the membership functions (MFs) to vary over their shape. In other words, the degree of uncertainty could vary along the abscissa of the MF shape. This could improve the modeling if there is both sufficient data and disagreement among stakeholders at that very detailed level – to the extent of varying uncertainty within the individual membership functions. For the SoS examples used, not much difference occurred when varying the entire MF shapes between trapezoidal and triangular. The modeling of the rest of the system, such as the strength of a capability contributed by an individual system, and how the capabilities are used together to achieve more capability at SoS level, would have to be improved as well to make this a worthwhile effort. Having the models all be at a roughly equal and appropriate level of fidelity is not necessary, but it avoids wasting effort.

The use of computing with words to find loci of commonality within stakeholder discussions when trying to establish the meaning of attributes and membership functions, is very similar to what is currently being done in 'big data,' and is very appropriate for the suggested modeling methods. Evaluating the relative value of individual SME inputs as suggested by Eggstaff, et al. could certainly be included in the modeling (Eggstaff, Mazzuchi and Sarkani 2014). Computing with words is at least 15 years old, but using it in conjunction with big data techniques is relatively recent. With many stakeholders and many conversations in a large SoS, finding principal components with the type of big data analysis used on Twitter, and computing with words approaches to define Type II fuzzy membership functions is quite feasible. Whether it significantly enhances the accuracy of the models depends on all the modeling components being done to the same level of rigor.

Intuitionistic fuzzy sets (IFS) are not a new addition to fuzzy logic, being at least 30 years old, but recently the concepts of interval valued IFS (IVIFS) and geometric aggregation operators have been introduced in a way that can make decision theory problems more realistic. These do not seem to be necessary or even helpful to the FILA-SoS approach, given the severe simplifications in the remainder of the model, but they could be used to improve the modeling of future, very complicated attribute functions, or if they required more complicated chromosomes to describe the architecture. Once again, due to the severe simplifications imposed by the binary meta-architecture approach, there do not seem to be advantages from incorporating relatively newer fuzzy logic topics such as topological fuzzy spaces, or continuity or separation characteristics of IFS.

2.4.3 Fuzzy Linguistic Analysis for Discovering SoS Attributes.

Mendel notes that there are numerous fuzzy approaches to allow 'computing with words' and to extract meaning even from the degree of our lack of knowledge to be included in the solution of a large variety of problems (Mendel 2013). Some problems with highly nonlinear relationships from many potential noisy inputs may be approached with fuzzy methods (Lin, et al. 1998). Li and Chiang (Li and Chiang 2013) introduce the concept of complex fuzzy sets, which even replace the 'if-then' rules of Mamdani fuzzy systems. Selva and Crawley use fuzzy sets to describe system attributes, along with artificial intelligence style rule based systems (up to hundreds of rules) to reason about potential architectures, but still largely see the result as binary – i.e., meeting requirements or not (Selva and Crawley 2013). They also recognize that the stakeholders themselves as part of the process, as well as being able to report results to them in easily understandable form, are important to the process. In systems acquisition, capabilities are usually the purpose of contractual requirements. Systems are traditionally acquired through contracts, and it is unreasonable to change the legal process. However, in acknowledged SoS, the capabilities are mostly already available, with only small changes potentially being contracted to add interfaces. The agreements to participate between system program offices (SPOs) and the SoS manager are usually not contractual but informal, such as in Memoranda of Understanding (MOU) or Agreement (MOA).

Many of the techniques mentioned above are more applicable to extremely large data sets, such as those of 'big data' in social media where sampling a huge population can detect trends and shifts in public opinion on the time scale of hours. Using them on a few dozens of SME opinions on engineering tasks or even the list of slightly more

numerous stakeholders discussed later seems inefficient, but they remain a viable approach for larger and smaller problems. Simpler, more basic techniques were used for this first modeling demonstration, leaving the obvious extensions to improved techniques for later (Agarwal, Pape and Dagli 2014). The attributes were selected and defined during weekly brainstorming sessions for a year among eight SMEs, with facilitation to determine consensus on fuzzy membership function shapes and bounds.

Much of the recent literature on fuzzy systems deals with treating uncertainty explicitly with Type 2 fuzzy systems. Type 2 systems treat the thickness of the membership function edges as an additional parameter in fitting a solution. There is a contention that adding parameters (and rules) to Type 1 fuzzy systems can be made equivalent to the extra degrees of freedom that Type 2 systems allow for describing solutions (Cara, et al. 2013). Several of these concepts were used in the definition of the membership functions and variable maps from real world variables to fuzzy variables here.

For the types and sizes of systems, capabilities, and missions involved in a typical SoS, there are substantial numbers of stakeholders and SMEs who would be interviewed, and numerous discussions to be undertaken over a wide range of facets of the proposed SoS. These discussions should provide a reasonable amount of data upon which to exercise the linguistic fuzzy analysis (Pape, Giammarco, et al. 2013). Wang and Zhang provide a possible approach to include the degree of uncertainty in the derived membership function definitions with Antonov's intuitionistic fuzzy sets (Wang and Zhang 2013). These concepts helped shape the discussion herein, but definitions were

kept as simple as possible to remain focused on the development of the overall method rather than fine points of possible improvements.

2.5 MULTI-OBJECTIVE FUZZY OPTIMIZATION

Satisfying the desires of many stakeholders over many attributes of the SoS is a multi-objective optimization (MOO) problem. A common method in the literature for solving a MOO problem is to use a genetic algorithm approach with a fuzzy fitness assessor as the chromosome sorter between generations (Pedrycz, Ekel and Parreiras 2011). Good chromosomes are more likely to be propagated to the next generation in most GA implementations. This technique solves multi-objective or multi-criteria problems by changing them into a single equation that can be optimized more easily. The combination of MOO with fuzzy approaches is discussed by Cara et al. (Cara, et al. 2013). Their problem was to fit surfaces with minimum error and minimize fuzzy rules while comparing Type-1 vs. Type-2 fuzzy sets. Several of their ideas are incorporated here, such as minimizing the number of rules in the fuzzy rule base. This has the advantage of making the architecture of the SoS easier to explain to stakeholders. (Type 2 fuzzy sets add uncertainty bands around the edges of the membership functions.) They also showed that Type 1 fuzzy systems are better in low noise (except for the input itself) situations, and Type-2 works better where the noise comes from the rest of the system. This effort uses the simpler Type 1 fuzzy systems, but an obvious extension to noisier, real world stakeholder linguistic inputs is possible. Wang and Zhang discuss incomplete information and weighted sets, but also include the concept of the penalty function as a more subtle method to push the fuzzy set solution off unwanted or infeasible solutions (Wang and Zhang 2013). A penalty function is incorporated in the FILA-SoS approach.

Sanz et al. (Sanz, et al. 2013) present the method used here of tuning the membership functions and rules to fit the data as the first part of their paper.

This method for selecting SoS architectures attempts to simplify and modularize the treatments of

- The SoS description - purpose, goals, constraints, etc.
- The definition of what is important to the stakeholders and how consensus is reached
- Selecting SoS attributes for evaluation
- Development process and funding within each system (cost and schedule are always a factor)
- Interactions between contributing systems when the SoS is fielded, and
- The negotiation between the SoS manager and the systems managers or SPOs to develop a realizable SoS.

A major effort was the segmentation of the models in an intelligent way, so that a variety of techniques could be tested with each other by 'dropping in' compatibly interfaced performance, evaluation, or display modules with different functionality. This was done by using well defined data files to exchange information between segments of the method. The modularity was also desired because it was not known which techniques would work best together, nor if different types of problems would require partially different approaches.

A fuzzy associative memory (FAM), normally generated by a fuzzy inference system (FIS), is a method of decision support that can satisfy, or select a compromise for, many objectives simultaneously. The multiple objectives may be thought of as

dimensions of a curve fitting problem. One common way to illustrate comparisons between approaches to a problem is by using a Kiviat chart (Microsoft Excel calls it a Radar chart), shown for example in Figure 2.9. The FAM is designed so that all possible combinations of attribute values can be ranked – this is the assessment at the SoS level. When created from the consensus stakeholders needs/desires through the method described in Chapter 3, the FIS is more justifiable than attempting to decide which of the two irregular polygons in Figure 2.9 is better. Genetic algorithms can explore such a 'space' very effectively, possibly without depending nearly as much on heuristics to simplify the solution approach. Minimizing heuristics is discussed further in section 3.9. When the space is the meta-architecture of a new SoS, the combination of 1) a fuzzy treatment for evaluating the attributes elicited by the method, 2) combining them to the overall assessment of the SoS architecture, and 3) the GA approach for finding a near optimum architecture, is a small step forward in the area of SoS engineering. The next sections discuss a fuzzy genetic approach to meeting some of the societal needs mentioned above.

Figure 2.9. Kiviat charts are sometimes used to show the satisfaction of multiple objectives

2.6 GENETIC ALGORITHM APPROACH TO THE PROBLEM

There are numerous genetic algorithm techniques (Fogel 2006) (Sumathi and Surekha 2010), from the very simple constant mutation rate on all chromosome members of the population, to random length transpositions at random positions, to sexual crossover at random positions, to variable size but 'gene' specific transpositions. In selecting chromosomes for reproduction to the next generation, techniques range from simple tournament selection of the best few, to roulette based 'higher fitness gives a greater chance of random selection (but not a guarantee)' for reproduction (Sumathi and Surekha 2010).

Some key drivers for the selection of a modeling approach are:

- The choice of representation of the problem
- The size of the domain
- Whether the gene components of the chromosome are possible (or worth it) to distinguish and treat differently
- The form of the fitness function used to select 'good' chromosomes from each generation.

The meta-architecture structure for the SoS problems addressed here was selected in FILA-SoS. With one small exception for the communication systems initialization, discussed in section 2.3.2, there are no privileged gene components in the SoS meta-architecture. One could treat the systems separately from the interfaces, as two genes within the chromosome, or certain combinations of systems' interfaces as a gene deserving special treatment. In many conceivable real SoS, this could be very useful and appropriate. However, this was not found to be necessary in this initial treatment. The

remaining driver to a solution is the choice of membership function shapes. The fuzzy logic system approach is well suited to the type of judgments made about 'good' SoS architectures (Pedrycz, Ekel and Parreiras 2011), but certainly not the only possible approach. In fact, other members of the FILA-SoS research group worked on several other methods of optimizing the architecture, including non-gradient descent methods and multi-level modeling.

Numerous programs or subroutines are published in C++ and Matlab for solving problems with GAs. Due to the fact that that the FILA-SoS established the file interchange format for the various elements of the overall approach to modeling the evolution of the SoS, a unique set of routines was coded for assessment and incorporated into a special purpose GA. These codes are included in Appendix B. Matlab Code. Most of the examples shown in Chapter 4 were computed using a hybrid of several GA techniques including tournament selection of the top 20% of the chromosomes in the population, replacement of the last 3 of those chromosomes with chromosomes from the lower ranked elements of the population, then replication of that top quintile portion 4 times: sexual crossover of random lengths of bits at random locations between quintile 2 and 3 was applied, transposition of random lengths of bits within each chromosome in quintile 4, and double the mutation rate of each bit in quintile 5 of the next generation population. Delta was specified mutation rate per bit, and also controlled the random location and length for crossovers and transposition.

Later in the research, a ranked roulette wheel selection algorithm was implemented in the GA. The literature suggested that this could be a better, faster, more effective GA approach (Kumar and Jyotishree 2012). This also demonstrated that the

fuzzy assessor approach was modular enough to be able to work with multiple GA approaches.

2.7 EVOLUTION OF THE SoS IN SUCCESSIVE WAVES

Another purpose of the FILA-SoS approach is to model the evolution of the SoS in successive steps called waves. After providing the suggested architecture to the other elements of FILA-SoS, negotiations are simulated between the systems and the SoS manager. The number of systems that choose to participate are typically less than all those invited. The realized architecture is assessed for quality, and plans for the next budget cycle (epoch) are implemented. Technology may change, new systems may come on line, and the opportunity to add systems, either from the same list or an amended list of systems occurs again in the next epoch. Participating systems from the previous epoch are protected; they have made the investments (and commitment) to participate already. These systems and their interfaces are protected from the random changes during optimization in the GA. After the GA operates through transpositions and mutations, any already participating systems and interfaces that might have been removed are replaced, so the evolutionary pressures occur only on the new systems for the next epoch. The protected systems are noted in input data to the GA.

2.8 SoS ARCHITECTING CHALLENGES

The first challenge is getting agreement on what constitutes an SoS. There is a continuing debate on this in systems engineering (SE) social media (such as the LinkedIn Community of Practice: Systems Engineering), over whether an SoS is merely a larger system, and even a debate over whether an SoS must be a complex system. This might

have been slightly less of a challenge if systems engineers could decide what systems engineering itself is. There has apparently never been an INCOSE International Symposium, or Workshop, over the past 25 years where the definition of the SE profession did not become a significant topic of conversation. There is a definition in the INCOSE Handbook, but many practitioners are dissatisfied with it; it gets at least slightly adjusted with each version release of the handbook. If the premier professional SE organization cannot satisfy themselves about what SE means, what hope is there of deciding what SoS Engineering is? There is a subargument that even if SoS may be slightly different from systems, there is no need to change normal SE processes because 'pure' SE is robust enough to take any differences into account.

This challenge can be answered by the authority of the US Department of Defense, an organization familiar enough with SoS to have a valid opinion, through their release of Systems Engineering Guide for Systems of Systems (Director Systems and Software Engineering, OUSD (AT&L) 2008). They describe a continuum of types of SoS, from tightly, centrally controlled (such as a military formation like a naval battle group) to extremely loosely controlled, voluntary, collaborative groups. They use the term 'virtual' for this end of the spectrum, but that term has taken on additional connotations since the Guide's publication, so that it requires clarification for this context. The Guide addresses many differences between what might have been considered a simple (but large) system, such as a weapons system in acquisition, and an SoS. The European Union is also firmly behind efforts (to the tune of millions of euros of research investment) to develop methods for handling SoS, such as through the COMPASS (Coleman, et al. 2012) and DANSE (Arnold, Boyer and Legay 2012)

programs. The European programs have the stated goal of becoming the premier practitioners of SoSE research and implementation in the world.

The next challenge is to attempt to describe and/or model an SoS in a succinct yet sufficient manner, especially to non-experts. SoS are almost always large and complicated, implying that it takes a correspondingly large amount of information to adequately characterize and explain them. Three key features of the proposed method help limit the problems inherent in this challenge:

1. The treatment is limited to only that type of SoS called ‘acknowledged (section 1.4),
2. The meta-architecture is limited to a binary participation model of systems and their interfaces, and
3. The purpose of this SoS analysis is limited in time and space to a single or at least a small range of scenarios.

The purpose of keeping the applicability of the method limited in this way is to see what one can learn from a simplified approach. Methods for collecting and organizing data for component systems, capabilities and interfaces are devised, with relatively simple models for performance and related ‘-ilities’ used to evaluate and compare arbitrary SoS architectures. This method is intended to be modular, so that competing or better models may easily be substituted. Other challenges for SoSE include crafting display techniques for architectures in different domains and evaluation criteria for SoS in those domains, displaying solutions, exploring sensitivity of the solutions to small perturbations, as well as summarizing relevant data for component systems in a concise presentation suitable for all stakeholders.

The International Council on Systems Engineering (INCOSE) initiated an SoS Working Group in 2012 to address some of the specific challenges of SoSE. Dr. Judith Dahmann, co-chair of the INCOSE SoS Working Group (WG), has consolidated seven 'SoS Pain Points 'over a period of several years, in conjunction with the National Defense Industry Agency (NDIA) SE WG, annual Conference on Systems Engineering Research (CSER), the Complex Adaptive Systems Conference (CAS), and the Trans-Atlantic Research and Education Agenda in System of Systems (T=AREA-SoS) (J. Dahmann 2014). While this research does not answer all the pain points in general, it does at least address some facet of each of them as shown below in Table 2.2.

Table 2.2. Proposed method's approach to SoS Pain Points

SoS Pain Points	**Questions**	**FILA-SoS Approach**
SoS Authorities	What are effective collaboration patterns in SoS?	First order undirected interfaces, but counts communication links as systems too
Leadership	What are the roles and characteristics of effective SoS leaders?	SoS Manager is the creator of the SoS vision & controller of a small budget for minor system changes; SPO managers negotiate for available/needed development funds
Constituent Systems	What are effective approaches to integrating constituent systems?	Assumed to be through information exchanges over communications links which are regarded as component systems
Capabilities & Requirements	How can SE address SoS capabilities and requirements?	An MBSE-like documentation approach to algorithmically account for system capability contributions to the SoS

Table 2.2. Proposed method's approach to SoS Pain Points (cont.)

SoS Pain Points	Questions	FILA-SoS Approach
Autonomy, Interdependencies & Emergence	How can SE address the complexities of SoS interdependencies and emergent behaviors?	Flexibility attribute asks for multiple contributors to each SoS capability Robustness attribute accounts for single missing systems Emergence arises from netcentric reward/penalty
Testing, Validation & Learning	How can SE approach SoS validation, testing, and continuous learning in SoS?	Costs, capability contributions, membership functions, and fitness rules may be varied for sensitivity analysis; Observed performance could be inserted into attribute evaluation algorithms to improve fidelity Wave model evolution can be explicitly modeled as systems/capabilities are added over time
SoS Principles	What are the key SoS thinking principles?	Fuzzy multi-objective optimization can handle large numbers of attributes Negotiations for realization of the suggested architecture Sensitivity analysis of input conditions, attribute membership function definitions and SoS assessment rule base

2.9 OTHER ARCHITECTURAL ANALYSIS METHODS

The Software Engineering Institute (SEI) at Carnegie Mellon University developed the ATAM, along with several related approaches such as Attribute Driven Design (ADD) to try to improve the quality of software programming (Nord, et al. 2009). It was primarily a way to get early expert review of the plans for large software projects to identify and prioritize any risky areas in the plan. This was in response to the

widespread and disturbing trend for large software projects to overrun their plans by very large ratios in both cost and schedule. At the same time this trend was becoming unmistakable, software was becoming the major component of most large and complex systems. This made it especially annoying to funding stakeholders, necessitating that "something must be done." ATAM, along with a number of other SEI initiatives were one result (Software Engineering Institute, Carnegie Mellon University 2015). They now have a Systems Architecture Tradeoff Analysis Method (SATAM), and an SoS Architecture Evaluation Method (SoSAEM), starting with Mission Thread Workshops (MTWs) that work through how the system will be used, that include the following steps:

> "An SoS architecture evaluation
>
> - uses outputs of the MTWs, including augmented mission threads and SoS architecture challenges
> - incorporates the expertise of a trained evaluation team and SoS stakeholders, including the SoS and system architects
> - probes architecture at the areas where the systems interact to identify risks
> - organizes the individual risks into risk themes that can be comprehended (and mitigated later) by program management
> - assesses the sufficiency of architecture documentation
> - identifies potentially problematic systems for focused follow-on evaluations using the specific augmented mission threads" (Software Engineering Institute, Carnegie Mellon University 2015)

However, the following line on their website notes that the method is “ready to be piloted,” i.e., not universally in practice yet.

ATAM is intended to be a high level, total system evaluation, very oriented to finding risky areas within the planned system architecture. Two other architecture evaluation methods are very closely related to ATAM:

1. Active Reviews for Intermediate Designs (ARID), also from the SEI, looks at portions of the architecture at milestone points as the system is developed. The key focus by highly experienced, typically outside, subject matter experts is again on the impact of architecture choices on quality attributes of the system, but ARID reviews are more focused, to only a portion of the total system, and only for the specific review being conducted (Software Engineering Institute, Carnegie Mellon University 2015).
2. Architecture-Centered Software Project Planning (ACSPP) is an approach where architecture documentation is provided to several experienced designers, they are allowed a limited time to prepare a plan to implement selected architectural elements, using the architecture documentation as inputs, identifying resources and time required. By comparing the resultant plans, differences in interpretation and clarity of the architecture description are highlighted by differences in the plans. Very high estimates of required resources or schedule are also interpreted as problem areas with the architecture descriptions (Paulish and Bass 2001).

Kruchten’s ‘4+1’ method for software architecting identifies four ‘views’ of the architecture: Logical, Development, and Physical. The ‘+1’ is at least one (or more)

scenarios for using the planned system (Kruchten 1995). There are numerous 'question' methods in the literature, where objective subject matter experts are asked to answer a list of questions from the architecture documentation alone. If the questions cannot be easily answered, the architecture description, if not the architecture itself, obviously needs to be improved.

This research are was disappointed in several ATAMs in which he participated. There seem to be a lack of depth of architectural detail, with an ad hoc nature to the architecture presentations. Criticisms offered by the participants seemed focused on only the most obvious risk areas within the selected architecture. Subtle or in depth analysis seemed quite beyond the group of objective subject matter experts (SMEs), perhaps because they were not well-versed in details of the architecture under discussion; they never approached the problem as a systems architecture review, but only commented in their narrow SME domains. This means that one must be careful to explain sufficiently when conducting the ATAM, facilitate helpfully, and keep the conversation at the architecture level. However, there did not seem to be a project commitment to do anything as a result of the risk areas identified in the ATM. Certainly, no thought was given to actually modifying the architecture as a result of the ATAM. Not every ATAM may be that superficial, but the process is certainly not immune to the superficiality observed. This is one of the criticisms levied on the ATAM approach, especially now that it has become institutionalized. It is common corporate practice for an ATAM to be required to allow a project to proceed, but a commitment to fix any highlighted problems as a hard requirement of the process snot always happen. Other methods identified above are also heavily dependent for success on the participation of highly skilled subject

matter experts with a systems approach. FILA-SoS assumes similar extensive SME participation in the generation of the attributes, evaluation models, and definition of acceptable ranges. It is the intent that this knowledge be better documented and more open than in some of the other evaluation methods. FILA-SoS seems to be following the dictates of ISO 42010 in this regard (IEEE S2ESC – Software and Systems Engineering Standards Committee) 2011).

There are few other 'architecture evaluation' methods per se. There are a growing number of architecture documentation and management methods. The US Defense Department has the DoDAF, which is oriented toward complex but still only a single system Program of Record (POR) (in the U.S. Congress acquisition terminology) style acquisition and design; Ministry of Defense has MODAF, NATO has an architecture framework, also oriented toward single (including complex) system procurements. DoDAF and MODAF have merged to become the Unified Profile for DoDAF/MODAF (UPDM). TOGAF, from the Object Management Group, is oriented toward enterprise architectures, not necessarily either systems or SoS, but possibly larger in scope than either. All these frameworks specify ways that the architecture description must be documented, with the sincere hope that any holes will somehow become obvious, and yet a further hope that they will be fixed, as well. The Architecture Analysis and Design Language (AADL) (now SAE Standard AS5506B) is another way of describing an architecture, concentrating on the interfaces as the most likely risk areas. This approach also helps point out holes in a proposed architecture that should then obviously be fixed. Steven Dam's proposal for a Life Cycle Modeling Language (LML) is designed to insure that entire life cycle concerns are included in the architecture description of a system

from early in its design, and is relatively easily extended to SoS (Dam and Vaneman, 2015). Although some modeling language tools are in the early stages of making architectural diagrams 'executable,' this is fairly experimental and certainly not widely used in practice. FILA-SoS has a module using Dori Dov's Object Process Modeling methodology (Blekhman and Dori 2011), and also Colored Petri Nets, both of which use the collected architecture data to create a discrete event model that may be used to test various hypotheses about the architecture, such as

- Does it have enough bandwidth?
- Is the latency of messages small enough under various usage conditions?
- Is there coverage throughout a shift, or a day, or a month, with this many units?

There are other discrete event modeling tools that can be used to examine the same types of questions with many other types of models than those built under the FILA-SoS approach. Recent European Union projects specifically oriented toward SoS include COMPASS – Comprehensive Modelling for Advanced Systems of Systems; DANSE – Designing for Adaptability and Evolution in System of Systems Engineering; DYMASoS – Dynamic Management of Physically Coupled Systems of Systems; AMADEOS – Architecture for Multi-criticality Agile Dependable Evolutionary Open System-of-Systems; and MONDO – Scalable Modeling and Model Management on the Cloud (COMPASS 2015) (European Commission's FP7 2015) (DYMASOS 2015) (MONDO Project 2015), all these approaches seem more oriented toward describing the architecture of, or managing an existing, SoS project, not so much as evaluating the architecture, although some do analysis of an existing architecture. There is a small portion of each of these efforts directed toward understanding where the project is now

(or will be in the near future). This is again, more of a hope that problems will make themselves obvious when describing the project in each of the above named method's terms than a direct evaluation of the architecture. The Systems Engineering Leading Indicators Guide from the International Council of Systems Engineering (INCOSE) (jointly with the Lean Advancement Initiative (LAI), the Systems Engineering Advancement Research Initiative (SEAri), and Practical Software and Systems Measurement (PSM)) also suggests to look at trends of metrics far more than at the absolute values of the suggested measures of health of an entire project (not merely at the architecture of the project). Another European suggested approach is the A3 size architecture overview method (A3AO) (Kooistra, Bonnema and Sko 2012). This is another method that highlights defining the architecture, distilled down to a single, moderate sized sheet of paper, with a relatively standard template of information to be included. Holes in the architecture description become glaringly obvious, one may hope, as noted previously.

Other architecture assessment methods have been proposed specifically for software, and many of these could be applied to SoS, with minor wording changes. Patterns have been proposed as a framework for evaluating aspects of software architectures. Patterns may be found in SoS, as well. Basically, they ask if certain problem patterns are present, and if they've been solved previously. If so, then patterns of previously successful solutions should be able to be deployed on similar problems in new architectures.

Functional dependency network analysis (FDNA) from Garvey and Pinto is another approach to architecture evaluation when the SoS is networked (Garvey and

Pinto 2009). It was primarily developed for supply chain types of analysis, which could be considered similar to an acknowledged SoS. This method of analysis is intended to find and understand risks in the supply chain. Many architecture analysis methods are oriented toward reducing risk in development or operation of an SoS.

SERC Research Task 108 on the SoS Analytic Workbench from Purdue University brings together several other methods for evaluating architectures. These include Bayesian Network analysis, Robust Portfolio Options, Approximate Dynamic Programming, and Stand-In Redundancy methods for evaluating SoS architectures. Another related method is Database Centric Architectures. All the above methods help expand the ways that architects think about, examine, analyze and select prospective architectures for complex systems in general, and SoS in particular.

All the above named approaches do help build better architectures. Most of them evaluate an architecture by finding risky places within it, or obvious (after being highlighted by application of the method) holes in the architecture. The SoS Analytic Workbench is more analysis focused than most. All the alternative methods focus attention on the architecture, and this is good. They do not 'evaluate an architecture numerically' as much as provide a path to improving it by suggesting areas to examine more closely. With the FILA-SoS approach, for each of the desired attributes, there is a documented model and score from the freely shared algorithms operating on each architecture instance. Even though the method maps those scores into fuzzy, qualitative measures, developed by analyzing a broad group of SME and stakeholder opinions, and the actual score is frequently on an arbitrary scale, or for only a short list of possible SoS scenarios, it provides something slightly more focused on the end result than some of the

evaluation methods in the literature. FILA-SoS adds value by combining the various attribute scores to an overall SoS assessment through the rule based fuzzy inference system, which can rigorously select between the Kiviat style representations in Figures 1-3. It also allows a more thorough exploration of the entire meta-architecture 'space,' as well as the ability to quickly assess any stakeholder's suggestions for improvement.

2.10 SCARCITY OF DOCUMENTED SoS EXAMPLES FOR STUDY

Institutional pressures make it difficult to find discussions of what works and what does not work in SoS. For one thing, such frankness is relatively rare. DoD examples of SoS typically:

- Do not follow the normal DoDI 5000 series management processes, so normal reporting is not always enforced, and therefore detailed records are unusually sparse
- Are not POR, so there is less than normal oversight by watchdog agencies
- Have relatively small budgets, or are started as pilot programs, not entailing the detailed oversight normally given to the bigger ticket items such as PORs
- Begin in an ad hoc way or as quick reaction efforts, so if they don't work, they are simply abandoned quickly for another ad hoc but more promising approach
- May be classified or have significant classified components, which make it exceedingly difficult both to record as well as to discover what happened in an accessible format

Commercial SoS efforts frequently fall under proprietary disclosure rules, which makes finding documented examples difficult in that arena, as well. Studies of failures

are infrequently objective, more commonly regarded as ‘the search for scapegoats;’ the participants surveyed after the fact frequently sense that agenda, and consciously or unconsciously, become reticent to share or even recall their part in the failure. Another barrier to finding good post-mortems on ‘problem’ projects is that those ‘lessons’ might be embarrassing to those most likely to know what occurred, so they frequently are not reported with full disclosure to protect the reputation of the organization or management chain. All these facets of SoS projects make it difficult to conduct accurate, reliable system case studies, or to find valid SoS lessons learned. The INCOSE SoS WG is actively engaged in finding SoS examples on which to do case studies, but even if they succeed, this will inevitably be a small sample.

Finally, it is often the case that no one really knows why a large project fails or succeeds. SoS are by definition complicated, therefore hard to understand, and normally have authority issues. The personnel assigned to the independent component systems have little motivation to understand the overall SoS architecture and purpose, hoping only to adequately fulfill their part (as they understand even that). The relatively few SoS engineers are normally sent off to other assignments as soon as an SoS failure is declared. No one stays around long enough to conduct a proper post mortem. It may simply be that an SoS success was an idea whose time had finally come, as much as one would like to ascribe to it a more helpful lesson of cause and effect, or even a rare example of excellent management. Another reason for success/failure might be that key stakeholder’s personalities made the project work (or not). It is a painful yet obvious truth that personalities have a great deal to do with success in complex projects. The next chapter

explains more of the unique underpinnings for applying the very simplified meta-architecture model to acknowledged SoS.

2.11 SUMMARY OF LITERATURE REVIEW

None of the concepts presented in this dissertation are particularly new or unique, with the exception of exploring the binary interface/participation meta-architecture and applying the fuzzy genetic MOO to SoS. The following concepts and different application of existing concepts do represent an addition to the growing body of knowledge in system of systems architecting:

- Development and demonstration of the method to create architecture based assessment models of the SoS that can quickly rank thousands of potential architectures
- Directly handling the ambiguity inherent in combining multiple systems into an acknowledged SoS, with its distributed authority and non-contractually binding requirements, with the fuzzy logic approach to attribute evaluation and SoS assessment
- Extending the application of DoDAF system oriented concepts to acknowledged Systems of Systems
- Using the upper triangular matrix representation of the binary participation in the SoS meta-architecture chromosome
- Treatment of the communication links between systems as elements of the SoS component systems

- Creating the concept of achievable and unachievable interfaces and connecting that to reward and penalty functions for the SoS netcentric improvement factor
- Using fuzzy, linguistic analysis on discussions with stakeholders to help define key performance attributes and explicitly handle the ambiguity in acknowledged SoS due to the sheer number of stakeholders and the lack of strong central control
- Providing a method to display the SoS interoperability architecture data including the concept of achievability
- Translating between fuzzy and real world representation of the attribute values through piecewise linear mappings to the membership functions
- Biasing the number of ones in the initial population of the genetic algorithm to explore a representative region of the meta-architecture space
- Applying the developed method across a number of SoS in different domains, addressing each of the seven SoS Pain Points to some extent

3. PROPOSED METHOD FOR DEVELOPING AN SoS EVALUATION MODEL

3.1 USE CASE MODEL OF THE DOMAIN INDEPENDENT METHOD

The method for developing an architecture evaluation model of an SoS is the same regardless of domain. Key features of the method are shown in the use case summary diagram of Figure 3.1. In this figure, dashed lines are for information; solid lines represent 'responsible for,' or active involvement; this portion of the effort excludes the Negotiate Achievable SoS use case The SoS manager is a key player, along with the SoS stakeholders, in forming a vision of the desired, acknowledged SoS capabilities. Information from potential component systems also contributes to the SoS vision. The vision of the SoS informs the model facilitator for exploring ways to model the desirable SoS attributes. This may include what fraction of the system capabilities the SoS will require, defining the meaning of the attributes and SoS missions in context, and establishing trade space limits to explore within the SoS meta-architecture. Other inputs include estimated costs for modification and operation of the systems within the SoS, which ideally would come from system stakeholders or SMEs, but usually start as estimates from the SoS manager. The modeler works with the model facilitator and various SMEs to develop attribute evaluation models that depend on the meta-architecture structure. These individual attribute evaluation models are combined through a fuzzy logic rule based system to assess the overall SoS. With this assessment tool, sample architectures represented in the meta-model may be evaluated for relative fitness as an entire SoS.

This fitness assessment tool is precisely what is needed by a GA to sort the better architectures within a mutating population of trial chromosomes searching out the meta-

architecture space. Sensitivity analyses can be run by the modeling team in consultation with the SoS manager. The consensus SoS design may then be presented by the SoS manager to the SPO managers for negotiation about any minor changes required to join the SoS. The documentation developed during the modeling effort is even more important for SoS explanation than for the legal and regulatory prescriptions of the DoDAF for official POR systems, because the SoS is outside the pre-existing design and training of the component systems. Results of the negotiations also need to be well documented, because SMEs may provide additional information to the negotiations, and stakeholders will want to know what capabilities their systems agree to provide to the SoS.

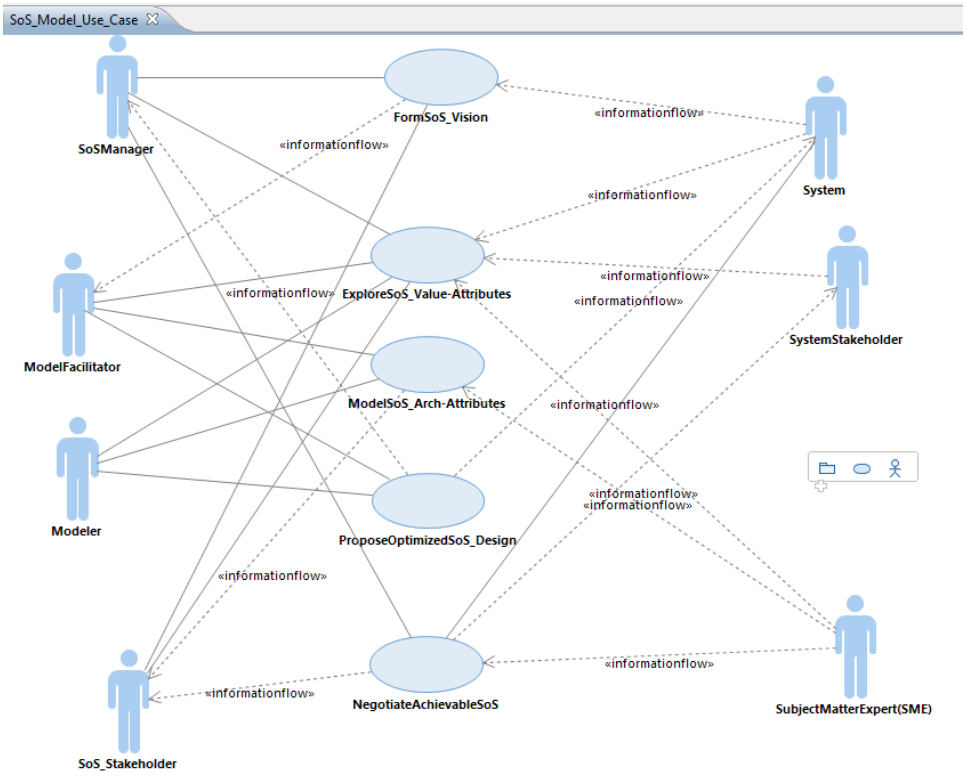


Figure 3.1. Use case diagram for developing an SoS Architecture

The list of data required, and the variable names used throughout this effort, for the generic SoS model is shown in Table 3.1. This is a simplified, binary model of the systems' presence or absence from the SoS, and the non-directed interfaces between each pair of systems.

Table 3.1. List of SoS and component systems' variable meanings within the meta-architecture

Name or description of variable	**Expression**	
Name of SoS:	sos	1
Number of potential systems:	m	2
Number of types of systems:	t	3
Names of system types:	$sys\_typ_i : i \in \{1,...t\}$	4
Number of component capabilities:	n	5
Names of component capabilities:	$sys\_cap_i : i \in \{1,...n\}$	6
Binary meta-architecture upper triangular matrix:	A_{ij} : $i \in \{1,\ldots m\}$, $j \in \{i,\ldots m\}$	7
Individual systems of the SoS	A_{ij} : $i \in \{1,\ldots m\}$, $j = i$, also sometimes written as A_{ii} , or simply A_i	8
Achievable interface	A_{ij} : $i \in \{1,\ldots m\}$, $j > i$, and $A_{jk} = 1$, $A_{ik} = 1$, $A_{ii} = 1$, $A_{jj} = 1$, $A_{kk} = 1$, where A_{kk} is any communications system	9
SoS main capability:	C	10
SoS performance in its large capability:	P_{SoS}	11
Component capabilities of systems:	c_{ij} : $i \in \{1,\ldots n$ capabilities$\}$, $j \in \{1,\ldots m$ systems$\}$ (binary matrix)	12
Performance of a particular system in its key capability:	P_i^{Ss} : $i \in \{1,\ldots m\}$, S_s is each system	13
Estimated funding to add an interface to an individual system:	FIF_i^{Ss} : $i \in \{1,\ldots m\}$, S_s is each system	14
Deadline for developing new interface(s) on a system:	D_i^{Ss} : $i \in \{1,\ldots m\}$, S_s is each system	15
Estimated funding for operation of all the participating systems during an SoS operation:	FOP_i^{Ss} : $i \in \{1,\ldots m\}$, S_s is each system	16

Table 3.1. List of SoS and component systems' variable meanings within the meta-architecture (cont.)

Name or description of variable	**Expression**	
Function describing the advantage of close collaboration within an SoS as a function of participating systems and interfaces:	$\mathcal{F}(A_{ii},\ A_{ij,\ j\neq i},\ ):\ i \in \{1,...m\},\ j \in \{i,...m\}$	17
Function for combining system capabilities into SoS capability C:	$C = \sum_k^n \sum_i^m A_{ii}\, c_{ki}$	18
Number of individual attributes the stakeholders want to evaluate the SoS over:	g	19
Attribute names to evaluate SoS architectures against (e.g., cost, performance, flexibility):	Att_k : k ϵ {1,…g attributes}	20
Number of gradations of each Attribute that become fuzzy Membership Functions (MF):	h_k : k ϵ {1,…k gradations within the attributes}	21
Fuzzy membership function names within each attribute (granulation = a, attribute = b):	MF_{ab} a ϵ {1,…h_k gradations}, b ϵ {1,…g attributes}	22
Fuzzy membership function boundaries (cross over points) for each of b SoS attributes:	$Bound_{ab}$ a ϵ {1,…h+1}, b ϵ {1,…g} a=1 is lower bound of universe of discourse, a ϵ {2,…h+1} is upper bound of $MF_{(a-1)b}$ because Matlab can't handle matrix subscripts of zero	23
Overall SoS performance in an Attribute	$(\sum_k^n \sum_i^m A_{ii}\, c_{ki}) * \mathcal{F}(A_{ii},\ A_{ij,\ j\neq i},\ )$	24
Total cost of developing and using an SoS	$TC = \sum_j^n \sum_i^m A_{ij} \mathrm{FIF}_i^{\mathrm{Ss}} + \sum_k^n \sum_i^m A_{ii} \mathrm{FOP}_i^{\mathrm{Ss}}$	25
Parameters for controlling the netcentric performance factor Increment per interface Penalty inc for unachievable Penalty decrement for achievable i/f	 Epsilon ϵ Penup Pendn	26
Parameters for controlling the GA: Mutation Rate Number in Population Number of Generations	 Delta P G	27

Figure 3.2 shows an alternate view of the method as a process flow with emphasis on the individual steps, without concern for who performs them.

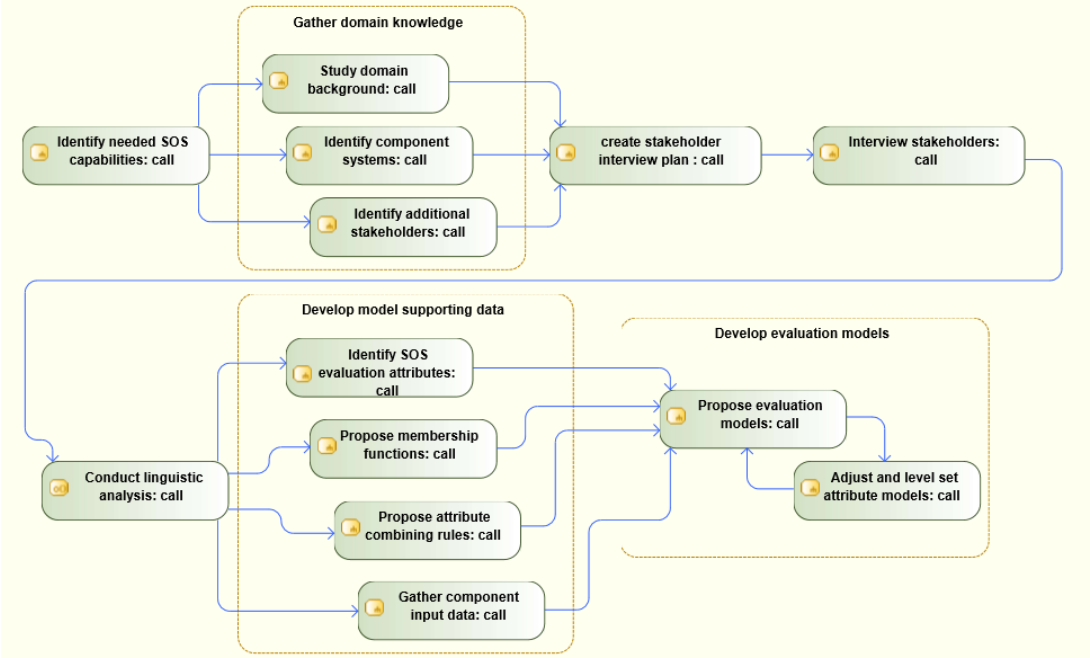


Figure 3.2. Domain Independent Process Method for SoS Model building

3.2 DOMAIN INDEPENDENT MODEL CREATION

The SoS model includes all the information available to it from the sources gathered from the participants identified in Figure 3.1, but it still must be cast in terms of the binary participation model of the meta-architecture.

The first step, regardless of domain, is to identify the reasons for the SoS and the desired capabilities. The SoS manager, and the facilitator, must always develop some background and vocabulary within the domain so that meaningful discussions may be held among stakeholders. At this point one can begin to create domain specific models of

development schedules, costs, performance, and other attributes to be used in evaluating an SoS architecture. The steps of the general method, however, are the same regardless of the domain of the model as shown in Figure 3.2. Many modeling approaches in the literature assume the architecture is already defined. This is similar to SE methods that assume the requirements are well defined – nice and clean, but neither realistic nor adequate. The DoDAF, Ver. 2.02, to its credit, begins at the proper place when it describes a domain independent six-step process for how to build an architecture model for a large DoD system:

1. Determine Intended Use of Architecture
2. Determine Scope of Architecture
3. Determine Data Required to Support Architecture Development
4. Collect, Organize, Correlate, and Store Architectural Data
5. Conduct Analyses in Support of Architecture Objectives
6. Document Results in Accordance with Decision-Maker Needs (ASD(NII) 2010)

This research extends the DoDAF system oriented model to SoS, adding detail on how to create, document and use a similar model building process for an SoS. This will form a basis to help designers and managers choose SoS architectures more wisely in the future.

The DoDAF viewpoints may be extended to the buildup of any SoS (military, civil or commercial) in nearly exactly the same way it is intended to be used to document the vision, plans, capabilities, and workings of a complex weapons system.

3.2.1 Establishing a Vision of the SoS. An SoS is by definition a group of independently capable systems, collaborating for a greater purpose, in other words, to deliver a larger capability. Within some range, systems may be present in varying numbers (or not at all) for a particular application of the SoS. The concept for the SoS must be articulated, captured, and agreed to among the stakeholders in relation to this variability in participation. Some SoS, after being developed, are on stand-by until called on to perform; others may implement a new capability that is operating all the time. The ideal SoS provides an acceptable range of capabilities over a broad range of compositions. Typically, the SoS manager (or management group) creates a vision statement to guide development of the concept for the SoS. The vision includes a high level description of the goals of the SoS, the potential types of participants and their capabilities, and the mission(s), threat(s), and a description of how the SoS arrangement will improve existing capabilities, or provide new ones. The architecture model of the SoS captures this vision but also provides the framework for decomposing the vision to manageable components as well as for building up the SoS out of legacy, new, or modified systems. The SoS manager must start with information such as that shown in Table 3.2 that corresponds to the DoDAF AV-1 Overview and Summary Information. (Corresponding roughly to Step 1 of the DoDAF 2.02 6-Step Architecture Development Process.)

Table 3.2. Example SoS evaluation model building questionnaire for creating an AV-1

Overarching Purpose Of SoS	A DoDAF OV-1 style description is often helpful; text should accompany it

Table 3.2. Example SoS evaluation model building questionnaire for creating an AV-1 (cont.)

Unique Value Of SoS	What makes it better than simply adding another legacy system				
SoS Measures Of Effectiveness	How will everyone know how good it is?				
Issues That Might Limit Effectiveness	Are changes of procedure necessary? Are there legal, regulatory, or bureaucratic impediments to the creation of the SoS?				
SoS Features That Might Greatly Increase SoS Effectiveness	Can changes in procedures help? What is the innovation?				
Desired Effectiveness	What would be considered really good, what's adequate, what's inadequate?				
Rom Budget: Development	As appropriate				
Rom Budget: Operations	As appropriate				
Desired Schedule	As appropriate				
Attributes Of The SoS/Range Limits For Fuzzy Evaluation	What might be 'tradeable' – Suggestions for fuzzy rules, e.g., is extra performance worth more budget? Is extra flexibility worth more? How much? Is lack of flexibility OK? etc.				
Capabilities Of Contributing Systems	How do they combine? Top level description of synergies				
Component Legacy Systems	Type/ Category	Capabilities	Time to Develop/ Equip	Costs $M- Dev and Ops	Notes (Incompatibilities, Constraints, Characteristics, etc.)
					etc.

An Operational View (OV-1) high level operational concept. The type of information that the SoS manager must have for the 'Vision of the SoS' is at least one example of how the SoS would be used (or a list of examples with all their context). The example must discuss expected participants in a rough picture (whether in graphics or text) of what the SoS will do in operation. Initial draft of this information may be summarized in one or two pages as shown in Table 3.2 for the All Viewpoint. This may be expanded to the OV-1 Operational Overview that describes how the system will be used in slightly greater detail. It can be a graphic with accompanying text showing the overall concept of use of the SoS as shown in Figure 3.3. Every term used in the descriptions is defined in the All View 2, the Integrated Dictionary (AV-2). Major component systems, data or resource exchanges, and effects are depicted iconically to present an overall, high level impression of how the SoS may be dispatched, controlled, employed, and recovered, for example, as shown in Figure 3.3. For an SoS, support is normally presumed to be supplied by the system operators in their continuing independent missions, unless significant changes are imposed by the SoS configuration. Major mission segments are shown are shown in the OV-1. The unifying SoS Integrated Dictionary (AV-2) is started with the OV-1, building outward so that all terms, components, activities, and interfaces are defined in one place.

Tracking the sources of definitions is more necessary for an SoS than for a system. Differences in usage of similar terms between component system stakeholders, model developers and operational users should be flagged in the AV-2 by noting multiple definitions for the same or similar terms within their proper contexts. This is significantly important in an acknowledged SoS because the nominally independent

component systems may have their own unique acronyms, terms or usages. The OV-1 establishes the scope of the SoS. A key component of most SoS is mission flexibility – the ability to pivot to different postures or missions as conditions change. A discussion of the range of likely activities of the SoS should be included in the textual explanation of the OV-1, or even as multiple graphics for different missions if that is part of the SoS charter. The OV-1 of an SoS must also include a discussion of priority between the SoS mission versus the original and continuing missions of the component systems. It should also include a generalized discussion of how deeply the SoS architecture will be allowed to control the component systems. That is, to what extent major interfaces enabling the SoS need to be controlled by (or at least communicated to) the SoS manager through the architecture, as well as where existing systems may continue control of their own configurations. (An extension of Step 2 of DoDAF 2.02 to handle the SoS.)

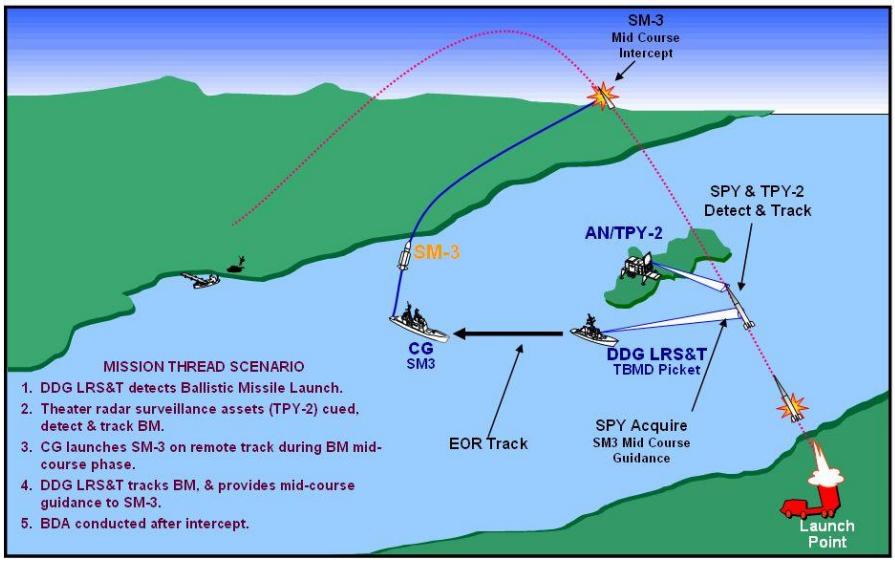


Figure 3.3. Sample OV-1 for ballistic missile defense (ASN/RDA 2009)

3.2.1.1 SoS Collecting descriptive domain information. Identifying the numerous stakeholders and their concerns, and gathering data about component systems and their missions are key parts of developing the required domain knowledge to build an SoS model. This process step is the same regardless of domain. The method is domain independent, but the data gathered is now domain dependent data. An initial rough level of knowledge is needed to allow a facilitator to make plans for stakeholder interviews. Identification of key discussion points and possible areas of tradable concepts within the early SoS construct are made at this point. However, until detailed discussions with the stakeholders are held, a facilitator must not jump to conclusions about what is valuable or tradable, nor even what the SoS framework looks like. Facilitated discussions with the stakeholders must draw out the following features of the SoS:

- Desired and composable capabilities, with expected or desired levels of performance
- Concepts of operation for the desired new SoS capabilities
- Likely scenarios for the employment of the SoS
- Key performance parameters, with expected or desired levels of performance
- Possible algorithms to combine component capabilities into SoS capabilities
- Shared (as well as conflicting) judgments about potential evaluation criteria for SoS attributes
- Relative ranking of, and expected values of, attributes of the SoS by groups of stakeholders

- Rough estimates of cost, schedule, and performance changes for required minor changes to existing systems to achieve desired SoS interfaces, or performance
- And to get an overall 'feel' for how the SoS might work in practice.

An important part of developing an SoS architecture is to define all the component systems' ownership, missions, and priorities in case some current mission capabilities must be 'hijacked' to support the SoS. Identifying all affected stakeholders is the second part of the facilitation exercise. Inside the Pentagon, this part of the effort is called identifying the coordination required to 'staff a position paper.' Since SoS normally include systems both from multiple domains, as well as across a range of stages of their life cycle, affected stakeholder identification requires careful and extensive coordination. As the stakeholders are identified, they should be placed in a hierarchy of command, tasking, and funding chains. This network is the basis of the Organizational Relationships Viewpoint (OV-4), which serves as an excellent template for SoS in any domain, not only military ones. In normal DoDI 5000.02 system development, this is nominally within one service, and most of the relationships are obvious. In an SoS, whether military, civil, or commercial, the effort to develop the hierarchies may require special attention and care to achieve successful coordination across major organizational boundaries (Director Systems and Software Engineering, OUSD (AT&L) 2008). Major concerns of each stakeholder must be discovered, recorded, tracked and updated over time, to aid in the coordination of initial tasking as well as changes to the goals of the SoS over its lifecycle. An ideal place for this information is in the OV-4 part of the DoDAF. Capability managers (or at least communities of interest) may be defined in the

Capability Taxonomy viewpoint (CV-2). These are cross-referenced and mapped in the Capability Dependencies viewpoint (CV-4) among the component systems and stakeholders. An ideal SoS would have a variety of ways to achieve each of its required capabilities, perhaps with varying efficiencies. Having only a single way to achieve a required capability is an exceedingly poor way to design an SoS; due to the independent nature of the component systems' missions, there is no guarantee that all possible systems will always be made available to the SoS. The concerns expressed in terms of the US DoD programs are equally applicable to any complex set of entrenched bureaucracies, such as companies in supply chains, divisions of corporations, or elements of intergovernmental enterprises.

The desired capabilities of the SoS, as well as those of the component systems must be carefully defined and accounted for both as a function of participating numbers of systems but also over time, as the SoS plans to mature. An ideal architecture should handle not only incremental improvements over time as capabilities evolve, but also a range of numbers of component systems. This accounts not only for technological improvements but also for the availability of systems. The number of systems can change on any particular day due either to logistic availability or to higher priorities outside the SoS. The attribute models of the SoS must be developed as functions of these variables. A SysML approach could allow parametric definition of capabilities and effectiveness to be explicitly built into the model. Other approaches may require additional math models, which ideally will be based on architectural data from the SoS model and the participation represented in the meta-architecture model.

3.2.1.2 Deducing attributes. Linguistic analysis of the stakeholder discussions ('computing with words') (Singh and Dagli 2010) allows one to deduce a set of attributes, potential membership function shapes, and rules for combining attribute values to create an overall SoS fitness evaluation. It may be necessary to iterate definitions of membership function shapes and rules to get a reasonable set that works together. Working together here means that the attribute measures do not overlap, nor correlate too well, among themselves (i.e., they are orthogonal, or nearly so). If they were duplicative, it would tend to give too much weight to a subset of issues, instead of optimizing over the broadest range of attributes.

Attribute characteristics and desirable ranges identified in the linguistic analysis are combined with fuzzy evaluation and a set of rules to derive a meta-architecture based, overall fitness value from the participating systems and interfaces. Level setting and model checking runs may need to be performed to insure the story is self-consistent. Then the model can be sampled across a range of percentage of ones for stakeholders' validation. These steps are as shown in Figure 3.4 and Table 3.1.

3.2.2 Understanding Stakeholders Views. A DoD acknowledged SoS is a very large, complex endeavor. SoS by definition create cross-functional organizations. They bring together functions that may have been built up through separate, large systems (and their program offices) that were developed over many years for many reasons, and only recently appear to have the potential to improve the effectiveness of a process or create a new capability by the joining together of these previously disparate systems. The new capability is highly desired, but not of overriding importance in the acknowledged SoS.

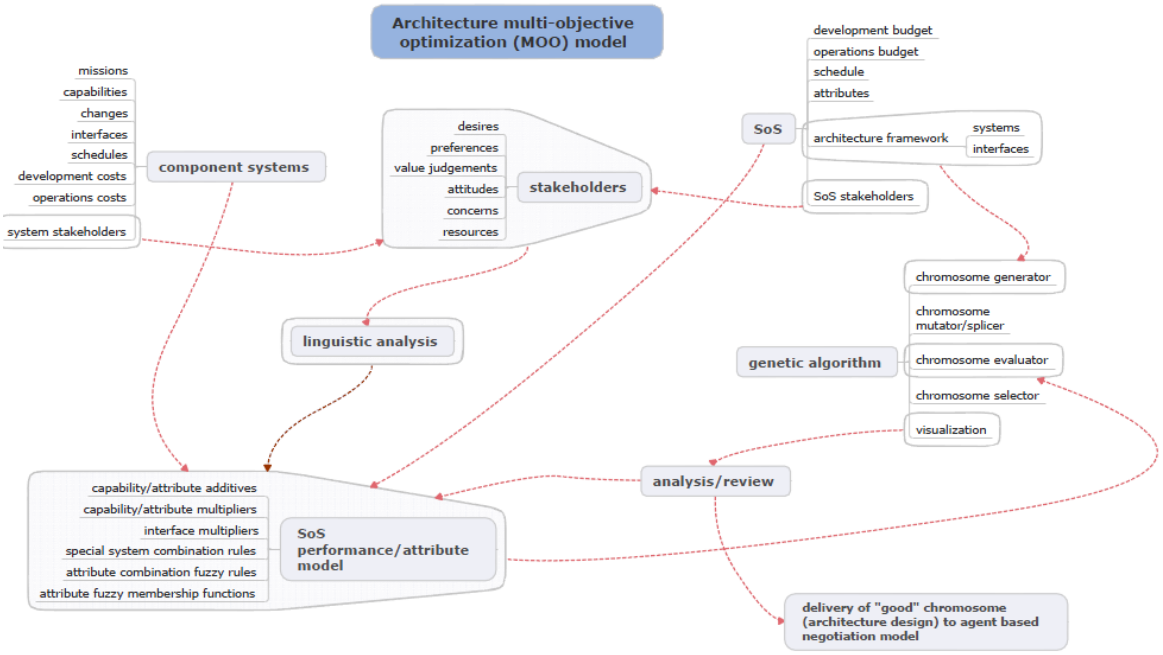


Figure 3.4. Data sources and analysis steps for applying the method

Many stakeholders are inevitably involved in an SoS. The stakeholders include at least the following recursive classes of interested parties:

- Component Systems (System Program Offices (SPOs) in the DoD or management agencies or corporations, and all the single system stakeholders that they represent)
- The SoS Manager or management agency
- Payers/funders (typically Congressional Committees, DoD, and Services for military systems, but also finance offices of other state or federal agencies, or CEOs of corporations)
- Congressional committees/watchdog agencies
- National or Theater Command Authority for military
- Users/beneficiaries of the SoS

- Operators of the SoS
- Competitors of the SoS
- Enemies/threats/targets of the SoS
- Allies of the U.S.
- Press/public opinion

A similar list could be made for other types of SoS in the civil or commercial domains. Occasionally individual stakeholders may be members of several groups simultaneously. Additional stakeholders may be professional organizations, industry groups, standards organizations, municipalities, rulemaking agencies, shareholders of corporations, charities, entrenched bureaucracies, unions, non-governmental organizations, etc. 'Due diligence' is the term for doing the work to identify the stakeholders of a proposed SoS, their degree of influence, and their level of concern about changes to their existing systems to make the SoS work.

3.2.2.1 Relationships to established decompositions: Task Lists, Joint Capability Areas, ISO Standards. When the domain is military, the Universal Joint Task List, the Service specific task lists, and the Joint Capability Areas provide excellent vocabulary for defining the missions and capabilities required for military tasks, independent of the systems used to achieve them (Joint Staff 2010) (j7jcaa@js.pentagon.mil 2009). This vocabulary of capabilities and tasks (activities in UML or SysML style modeling) is aggregated in the SoS AV-2 and model behavior definitions, so that each time that a term, word or concept is used in the architecture, reports, or performance models, it is consistent and clear to every stakeholder or participant. Other domains than military typically have manuals, corporate, industry or

government standards, scholarly, or professional guidance documents, or even textbooks to provide this background of vocabulary and definitions. The ISO-10303 series of standards is another source of guidance, particularly AP233, Systems Engineering Data Representation. In fact, there are usually so many possible sources that it is highly advisable to maintain source tracking within the AV-2, with priorities assigned to each source to prevent confusion when a term is overloaded by multiple definitions depending on context.

The DoD task lists contain suggested very high level definitions of measures of effectiveness for evaluating the performance of the capabilities. These are potentially valuable sources for determining membership function shapes and edge values. These are typically 'improved upon' for specific system solutions, but they serve as an excellent starting point for drafting evaluation criteria for the SoS, especially in performance. For the first pass through a fuzzy analysis, the membership function shapes are not too important; triangular or trapezoidal shapes work well enough to get started. At the preliminary stage of analyzing choices with crude, initial models, it is more important to get the terminology, ordering, and trade space rules agreed to among the stakeholders than to have highly accurate membership function shapes.

Other '-ilities' models may contribute to SoS attributes – reliability, availability, affordability, survivability, flexibility, adaptability, agility, ability to be redirected, autonomy, precision, among many others (Mekdeci, et al. 2014) , may be useful in evaluating characteristics of a particular SoS meta-architecture. SoS attributes should be created through reasonable extrapolations of the component systems' capabilities in each area, with a small improvement factor for self-coordination. (If the SoS has no advantage

over the simple sum of component systems' capabilities, then there is no need for the SoS – simply send more systems to do the task.) By the time one has defined the vision, capabilities, stakeholders, components, and measures of effectiveness, there should be enough of a basis to decide what additional data will be required to develop the architecture evaluation models. (Step 3 of DoDAF 2.02.)

3.2.2.2 Capability improvement of a proposed SoS. The concept in the FILA-SoS for the buildup and even emergence of capabilities within the SoS is that capabilities are brought to the SoS basically intact by the component systems as currently existing. Typically, the SoS improves the sum of the individual component system capabilities by a change in the way they work together to provide some unique or even greatly improved capability. Assume the interfacing of those systems together in a new way can be made to improve performance by a small multiplier for each connection. This is a typical approach introduced as the concept of netcentricity by Alberts, Garska and Stein in the late 1990s (Alberts, Garstka and Stein 1999). This is equivalent to the small change in performance for each used-achievable interface. It is a greatly simplified notion to regard the performance improvement to be a simple exponential function of the number of interfaces; there is undoubtedly a plateau effect on the lower end whereby a minimum number of systems must be interfaced to be able to see the effect. On the high end there is no doubt also a limit to improvement by the introduction of the concepts of information overload, latency, and bandwidth limits. The simplifying assumption that more interfaces is better is nevertheless quite reasonable over a broad range between the two extremes, especially since it is limited to a small fraction for each interface.

3.2.2.3 Decomposition of capabilities to functions and logical views. The high level capabilities described in the AV-2 and OV-1 can be decomposed to lower level actions and/or functions allocable to the potential component systems. This continues iteratively, exactly as in normal/standard systems engineering, until both a functional hierarchy and behavioral description can be attributed to component systems. Some systems may need upgrades to be compatible with the SoS architecture. The phasing and organization of the capabilities must be agreed to by both the systems and the SoS manager, with performance, funding and schedules. The time phasing of capabilities development is shown in the Capability Phasing Viewpoint (CV-3). If some systems' capabilities were to be ready before others and they could be used together, the timeframes would be noted and this would become a Capability Vision Viewpoint (CV-1) that shows how the deployed capability is built up (ASN/RDA 2009). Mapping capability development to operational activities is shown with the Capability to Operational Activities Mapping Viewpoint (CV-6). If some activities are not possible without the developing capabilities, then there will be some operational changes over time, as well. Some functions may be logically grouped because they can be reused to support other missions; some might be grouped because they are unique to the SoS mission and configuration. Training and tactics may have to be developed to use new capabilities, or even to get the systems to work together operationally if the systems don't already do so in existing, joint missions. These constraints may be shown in several of the capability views, but especially in the Capability Dependencies (CV-4) and Capability to Organizational Development (CV-5) viewpoints.

The decomposition of capabilities to functions, and the aggregation of functions to higher levels of abstraction, eventually to capabilities, are inverse processes. Sometimes it is easier to decompose downward, other times it is easier to aggregate upward. This depends on what information is available when one starts the process. The important point is to fill in the Capability Taxonomy Viewpoint (CV-2), so that it is complete and makes sense to all stakeholders (or at least is accepted by all) as the operative definition for the SoS. The capability taxonomy is a subset of the Integrated Dictionary definitions, with the addition of the item's location within the hierarchy. Naturally, it is best to think through the implications of the definitions for the whole lifecycle of the SoS. This also implies that the vision should be sufficient to sustain a lifecycle view for the SoS, not merely the initial use of it. In practice, this sufficiency of vision is rare.

Many SoS, in spite of being complicated arrangements, are also started as quick reaction responses to environmental changes. Therefore, many SoS are short time frame exercises. "Make the required changes quickly, and get them deployed!" is the prevailing attitude in this case. In this extreme, there is scant thought given to planned upgrades, phased deployment, or building for growth. Here, all the changes or new interfaces need to be developed in one time period (usually a budget period, called an epoch in FILA-SoS). When there is planning for development over several epochs, the 'in-work' interfaces are regarded as not participating until their delivery epoch.

The development of the Architecture Views must be an ongoing, continually updated process extending throughout the program life cycle. The simplest cocktail napkin draft to the most detailed, data base driven, multiply approved, fully vetted,

graphical interface control definition should be documented within a "model," as the single source of data. Many of the remaining DoDAF viewpoints can be derived from basic Operational Activity Model Viewpoints (OV-5b) activity diagrams if they contain both activities allocated to swim lanes (denoting the various participants/elements/actors) and sequenced data flows between elements. This is an addition to the basic (minimalist) definition of an activity diagram, but adding these two items is an important step in defining how the SoS will operate. One can vary the amount of back up text residing in each object in the model. This is dependent on the amount of detail required and available at each stage of the architecture development. However, the AV-2 works most brilliantly if two conditions are fulfilled: all participants assiduously define their terms in it, and a facilitator continually edits its contents for clarity and consistency. Consistency is sustained if the rest of the documentation uses the AV-2.

An architecture of the SoS will exist, whether or not it is defined, planned, or understood. It will be a far more useful architecture (and a better SoS) if the architecture is developed intentionally, and well documented. That the documentation might be in a standard, organized framework, and maintained throughout the life of the SoS in a central repository, could make it useful to new hires, visitors, and the engineers and managers attempting to upgrade, maintain, or use the SoS in the future. (Step 4 and 6 of DoDAF 2.02.)

The Integrated Dictionary (AV-2) is the authoritative source for definitions of all elements of the architecture and program descriptions. All acronyms, terms of art, and important concepts must be defined there, and the source of the definition is maintained to give context for understanding arcane, duplicative, or cross program usages (frequent

occurrences in SoS). Example architectural element definitions found in an AV-2 are shown in Table 3.3.

Table 3.3. Example AV-2, Integrated Dictionary

Phrase	Acronym	Definition	Source
Computing Infrastructure Readiness	CIR	Provides the necessary computing infrastructure and related services to allow the DoD to operate according to net-centric principles. It ensures that adequate processing, storage, and related infrastructure services are in place to respond dynamically to computing needs and to balance loads across the infrastructure.	DoD IEA v2.0
Concept of Operations		A clear and concise statement of the line of action chosen by a commander in order to accomplish his mission.	Std I/F UCS Nato STANAG 4586-3
Conceptual Data Model	DIV-1	The required high-level data concepts and their relationships.	DoDAF 2.02
Condition		The state of an environment or situation in which a Performer performs.	DoDAF 2.02
Confidentiality		Assurance that information is not disclosed to unauthorized entities or processes.	DoD IEA v2.0
Configuration		A characteristic of a system element, or project artifact, describing their maturity or performance.	INCOSE Sys Eng Hndbk v3.2.1

3.2.2.4 Conducting analyses of SoS behavior. The SoS manager and development facilitator must at this point be doing some mental estimation of where the required SoS capabilities could be obtained and for what cost. They must be designing questions to elicit both responses and thought from the stakeholders about what could be of value in building the SoS. The stakeholders extend both up and down the chain of

responsibilities with the SoS manager in the middle. Are there potential multiple sources for most required capabilities? Are there new ways of putting pieces together in different ways to accomplish necessary tasks or functions? Do new technologies allow for anything more easily that previously envisioned? If something did work in a new way, how much better would it be? What functional relationships could be described to evaluate the SoS? What ranges of values of performance would be outstanding, pretty good, acceptable, poor, or awful? Answering these questions will allow models to be built that will allow new designs of an SoS to be evaluated. (Step 5 of DoDAF 2.02. Step 6 is documenting the viewpoints in a self-consistent model, which is done during all the previous steps.)

3.2.3 Review of the Method Steps. Yet another way to look at this model development process is shown in Figure 3.5, using the binary participation meta-architecture model as a starting point. A vision of the SoS, facilitated stakeholder discussions, produces a plethora of linguistic terms and definitions. Linguistic analysis of these discussions may be used to distill the SoS attributes that are important to the stakeholders. Linguistic analysis also may be used to establish ranges of values for the attributes that are considered excellent, good, or very bad, as well as the strength of the stakeholders, feelings about each of these ranges. The modeler gets to play a role at this point by writing trial algorithms that work on the meta-architecture to deliver an initial trial measure of each attribute. These measures should depend significantly on the meta-architecture, because they are used to evaluate the goodness of the architecture of the SoS. Given this information, establishing membership functions to fit the fuzzy evaluation measures is relatively easy. Rules for combining attribute valuations are also

developed from stakeholder interviews and discussions. The rules are embodied in a Mamdani fuzzy inference system or fuzzy associative memory in the form seen in Table 3.5 in paragraph 3.6. The measures are used to improve the selection of the SoS architecture within the genetic algorithm approach. The optimized architecture is then proposed for implementation and negotiation between the component systems and SoS manager. The negotiations require a reasonably good starting point to have any chance of success, and that is what this research is designed to provide – the starting architecture for the agent based modeling part of the problem. The system negotiations are the key to getting a realizable, implementable architecture for the SoS, because the systems cannot be forced into the SoS in the case of an acknowledged SoS.

It is important to devise a method to visualize how the component systems' capabilities ($\mathbf{c_i}$) of various architecture instantiations come together to create the SoS capabilities ($\mathbf{C}$). This helps during the level setting exercises, but is vital to describing both the approach and the results to stakeholders as well. Finally, there must be clear explanations of the limitations of the modeling approach. The numerous simplifications mean that the model is not likely to match reality very well in detail; the best this model can do is match in terms of broad, general trends in comparing high-level architectural impacts between different approaches to constructing the SoS.

For every SoS there will be requirements for component system and capability descriptions. Capabilities of each system are denoted by $\mathbf{c_i}$, and the way those capabilities are combined into the SoS capability $\mathbf{C}$ must be described and captured in the model. When the information is gathered and organized, then the domain specific model

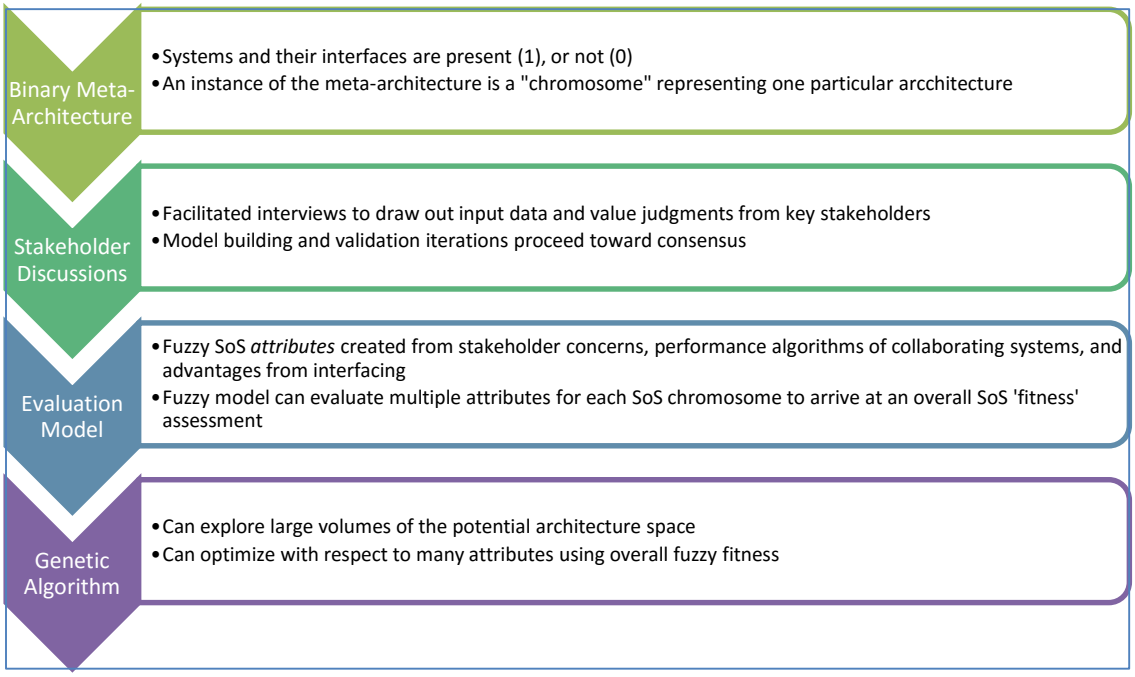


Figure 3.5. How the steps of the method result in a good SoS architecture

is described, but the fact that there must be some way to build up the required capability as a module in the model is domain independent. For our meta-architecture, there may be sets of capabilities from each system, a combination algorithm to describe how the SoS capability is built up from the systems, and costs for development of either new capabilities or interfaces, with schedules and costs for operations of the systems in the SoS. More detailed models could be used, for example if the cost of discovering and codifying new doctrine or tactics, and training in the new configurations is known or can be estimated. Other desirable attributes and ways of measuring and combining them may be discovered during the stakeholder discussions; these may be added to the SoS evaluation, but there must be some process such as this regardless of domain. Initial draft runs of the model may also lead the modelers to changes and improvements in the model

modules. This method of discovering the domain dependent data that goes into the model is domain independent. It should work for any domain.

3.2.3.1 Choosing the SoS key attributes. The facilitators and model builders need some basic knowledge of the domain of the SoS. Without that, they will not be able to ask intelligent questions of the stakeholders and SMEs. Conversely, if the facilitators know too much about a domain, they may unconsciously pre-select a solution, biasing the way they ask questions. An example questionnaire form for directing the interviews with stakeholders is shown in Table 3.2. This is only a very high level starting point; it should be adjusted for any specific SoS application.

Questions such as those in Table 3.2 are intended to elicit from the stakeholders the key attributes (or key performance parameters (KPPs)) that they care about for the development and use of the SoS. The questions are asked from the point of view, and with the intention, of developing relatively simple evaluation algorithms that depend strongly on the participating systems and their interfaces and how they interconnect in operation. At the initial stage of development, these algorithms may be fairly approximate, using rough estimates. The goal is to have some kind of broadly achievable architecture with which to begin the analysis leading to negotiations between SoS manager and individual systems for an agreed to SoS. It is fully expected that individual systems' performance, cost, schedule and other attributes would be adjusted during negotiations. On the other hand, attribute evaluation algorithms are designed to be modular, so that if better models become available, they may be substituted in at any time. Well documented, traceable information trails are invaluable when reviewing,

improving, correcting, or extending the models (or the SoS itself). These documented traces are well described by DoDAF style views.

3.2.3.2 Visualizing domain model data. There is a great deal of information potentially available about the components of the SoS – each system may be complex in its own right. The architect must find a better way to share SoS data with analysts and stakeholders than dozens or hundreds of columns of numbers. Color coded and textured graphs, multiple and rotatable viewpoints into multi-dimensional data, slices, smart filtering, correlations, time series analyses, and animations may all be used to aid the understanding of the very large sets of data produced by SoS modeling (Yi, et al. 2007). Both large scale trends and significant but tiny artifacts in the data must be easily and quickly discoverable in the way the data is conveyed to reviewers. One needs to be careful of the color palette chosen to display results, since different display or printer devices may represent them differently, sometimes in surprising ways. Some in the audience will usually be color blind to various degrees, as well. One must be careful not to assume that color coding data artifacts makes them obvious to others. (For those interested, the website http://www.vischeck.com/examples/ simulates for people with normal color vision the way colorblind people see, and suggests alterations to color palettes that allow more people to see an image in a similar way).

3.2.4 Architecture Space Exploration. This modeling method uses a genetic algorithm (GA) approach to explore the architecture space. A population of chromosomes is evaluated and sorted to select the better ones for propagation to future generations with genetic modifications. The ones and zeroes in the chromosomes are generated at random during the first generation. This is normal GA procedure. However,

to avoid getting an average of 50% ones in the entire initial generation of chromosomes, a bias is applied to the random number generator so that the probability of any bit in a chromosome of that generation being a one depends on the chromosome's position in the population. The number of chromosomes in a population for one generation of the GA is variable. Typically, a few tens to a few hundred chromosomes are used in the population in each generation. For the initial generation, chromosome number 1 has only a few ones, with mostly zeroes. The last chromosome in a population has mostly ones with only a few zeroes. Typically, low numbers of ones in the chromosome (meaning participating systems and/or interfaces) is associated with lower cost and lower performance. The other attributes could be better or worse, depending on their definitions. Higher numbers of participating systems and interfaces are usually associated with higher performance and higher cost (equation 24 and 25 in Table 3.1). Costs/affordability and overall performance are almost universally necessary SoS evaluation attributes. Since there is normally a desire for higher performance and lower cost, one hopes for a sweet spot between the extremes, where there is adequate performance, and adequate affordability (nearly the inverse of cost) as well as acceptable values of the other attributes.

A key feature of the method is to do an exploration of the architecture space with a few hundred or a few thousand sample chromosomes, which cover a large range of participating systems and interfaces. Each of the chromosome's attribute evaluations is plotted against the membership functions for that attribute. The membership function shapes and/or the algorithms for evaluating the attributes may need to be adjusted several

times in an iterative process that may include discussions with stakeholders to arrive in acceptable SoS model. This is explained further in section 3.8

The meta-architecture and associated data model being proposed so far contains many features that mimic real-life:

- There may be multiple copies of the same system
- There may be slight differences between the otherwise similar systems
- Each system may have multiple capabilities
- There is a minimum number of component capabilities required to make up the SoS capability.

If a proposed architecture does not have the minimum capabilities, a penalty is tacked on to its performance, to enhance the chances of discarding its chromosome in the fitness comparisons at each generation of the genetic algorithm. No population member or bit position is pre-selected for discarding before evaluating it for all attributes.

3.3 INDIVIDUAL SYSTEMS' INFORMATION

3.3.1 Cost, Performance and Schedule Inputs of Component Systems. The models used here treat the cost of developing a new capability or adding an interface separately from the cost of operating the system during a deployment of the SoS. In real life these costs are potentially paid for from different categories of funds, such as acquisition vs. operations budget lines in DoD, or current vs. future funds. The development cost is normally a one-time cost, while the operations cost of the system is a continuing cost each time the SoS is called into action. Performance enhancement is normally on the order of a few percent for the modification requested to fit into the SoS. For adding an interface, it might require a new radio and antennas to be installed on a

vehicle, or extending the software database of messages that can be handled by an existing system on a vehicle. There can be significant costs for a minor modification to accomplish retesting of functions that might be affected by any changes to fielded systems (called regression testing), in addition to testing of the change itself. Whatever the change, in addition to time to develop and test the change, the system hosting it will be 'down for maintenance' during the installation of the change. The time to develop, install and test a change is the development time. This is generally one epoch, or time period, in the wave model described in Chapter 1. If a capability already exists, such as ability to use a specific radio on a platform, the development cost and time for that system for that capability will be zero in the domain input data. However, development of the other end of the interface on a different system may still be required and will count toward the cost of the interface. Some complex modifications might take two or more epochs to develop. In this case, since the development is not complete at the end of the first epoch, it is as if the system chose not to participate, because it delivers no capability yet. However, one is still spending funds on that development, for which one receives nothing until the development is complete. The bottom of Table 3.2 shows a simplified template to start gathering the estimated individual system cost and performance data. Figure 3.6 and Figure 3.7 show a typical arrangement of the input data required to perform the individual attribute evaluation model calculations. Figure 3.7 is in the latest graphic user interface format. Columns have been added for the protection of existing systems and for the negotiation behavior; the remainder of the columnar information is

the same. Heading information has been modified to allow a few more adjustable settings in the data input file.

3.3.2 Membership Functions. Membership functions (MF) map the fuzzy values to the real world values and show the fuzziness of the boundaries between the granulations or grades within each attribute. The Matlab Fuzzy Toolbox has a number of built in shapes for membership functions. Triangular, trapezoidal, and the Gaussian smoothed corners of trapezoidal shapes are available among others; only the Gaussian rounded trapezoidal shape shown in Figure 3.8 was used in this analysis. It is very common when evaluating large projects to have a band of acceptability for each grade in each attribute. A familiar example is the Contractor Performance Assessment Reporting System (CPARS). It assigns one of five colors to a series of common measures of project status. It is also common to have multiple reviewers provide a grade in each area, which is then averaged to get the final grade (Department of the Navy 1997). This is intended to avoid the issue of unconscious bias or error of interpretation of the data by a single reviewer on a borderline issue. This process is very similar to the mathematically more precise fuzzy logic process. Other MF shapes show similar characteristics, but the nonlinearities in the output surface display the concepts better with the slightly rounded MF shapes shown in Figure 3.8. All the variables in the Fuzzy Tool Box are scaled the same way. In real space, a further scaling is required to the individual variables. The MFs cross each other at the 50% level between each of the numbers in the granularity scale from 1 to 4. For this fuzzy inference system (FIS), 1 = Unacceptable, 2 = Marginal, 3= Acceptable, and 4 = Exceeds (expectations) for each attribute: Performance,

Name	SAR					
NumSys	29		com1	26		
NumCap	10		attr	4	mfnum	4
SysNo	**Type**	**Capability**	**I/FDevCos**	**OpsCost/I**	**Perf**	**DevTime**
1	Cutter	7	0.03	2	12	1
2	Cutter	7	0.03	2	12	1
3	Helicopte	6	0.1	2	20	1
4	Helicopte	6	0.1	2	20	1
5	Aircraft	8	0.1	5	10	1
6	Aircraft	8	0.1	5	10	1
7	UAV	1	0.1	0.1	7	1
8	UAV	1	0.1	0.1	7	1
9	UAV	1	0.1	0.1	7	1
10	UAV	1	0.1	0.1	7	1
11	UAV	1	0.1	0.1	7	1
12	UAV	1	0.1	0.1	7	1
13	UAV	1	0.1	0.1	7	1
14	UAV	3	0.1	0.1	7	1
15	UAV	3	0.1	0.1	7	1
16	UAV	3	0.1	0.1	7	1
17	UAV	3	0.1	0.1	7	1
18	Fish Vesse	3	0.03	0.5	4	1
19	Fish Vesse	3	0.03	0.5	4	1
20	Fish Vesse	3	0.03	0.5	4	1
21	Fish Vesse	3	0.03	0.5	4	1
22	Fish Vesse	3	0.03	0.5	4	1
23	Civ Ship	7	0.05	2	8	1
24	Coord Ctr	5	0.05	0.5	5	1
25	Coord Ctr	5	0.05	0.5	5	1
26	Communi	10	0.02	0.03	1	0
27	Communi	10	0.02	0.03	1	0
28	Communi	10	0.02	0.03	1	0
29	Communi	10	0.02	0.03	1	0

Figure 3.6. Example domain data input for 29 system SAR SoS

SoS Description	ISR						
Total Systems in SoS	22		probtyp	ISR			
Number of Characterist	6		linearin	1			
Number of System Type	22						
Max negotiation rounds	4						
	ProtectNeg	Behavior	Perf	OpsCost	IFcost	DevTime	
fighterA1	0	2	10	10	0.2	1	
fighterA2	0	2	10	10	0.2	1	
fighterA3	0	2	10	10	0.2	1	
RPA1	0	2	10	2	0.4	1	
RPA2	0	2	10	2	0.4	1	
RPA3	0	2	10	2	0.4	1	
RPA4	0	2	10	2	0.4	1	
U-2	0	2	3	15	0	0	
DSP	0	2	8	0.1	1	1	
fighterB1	0	2	15	10	0.7	1	
fighterB2	0	2	15	10	0.7	1	
fighterB3	0	2	15	10	0.7	1	
JSTARS	0	2	40	18	0.1	1	
ThExp1	0	2	10	10	2	1	
ThExp2	0	2	10	10	2	1	
ConUS	0	2	15	0.1	0.2	0	
CmdCont1	0	2	12	2	1	1	
CmdCont2	0	2	12	2	1	1	
LOS1	0	2	10	0.1	0.2	1	
LOS2	0	2	10	0.1	0.2	1	
BLOS1	0	2	10	3	0.5	1	
BLOS2	0	2	10	3	0.5	1	

Figure 3.7. Updated input data format for characteristics of the component systems

Affordability, (Developmental) Flexibility, and Robustness. There is no requirement that the scaling be the same for different attributes. In fact, the Matlab Fuzzy Tool Box allows the MFs to be scaled to real values, and it might have been clearer to use that facility, but the graphical user interface (GUI) for changing the values is rather tedious, so a method to apply the scaling outside the GUI was developed. The process of

translating real values to fuzzy values is called fuzzification or fuzzifying. Multiple criteria are combined through the rules in fuzzy space, and the output fuzzy value is de-fuzzified to a crisp value for the SoS assessment. In this fuzzification scheme, values are rounded to the nearest integer value for each fuzzy gradation. In the example in Figure 3.8, a fuzzy value of 2.35 would fall on the sloping line for Marginal membership at a value of about 65%, higher than on the line for membership in Acceptable, where it is about 35%. The crossover points between membership functions are fixed at the half integer point in fuzzy space, but need not be mapped linearly to real space. The next section discusses how the real values are mapped to the fuzzy scale.

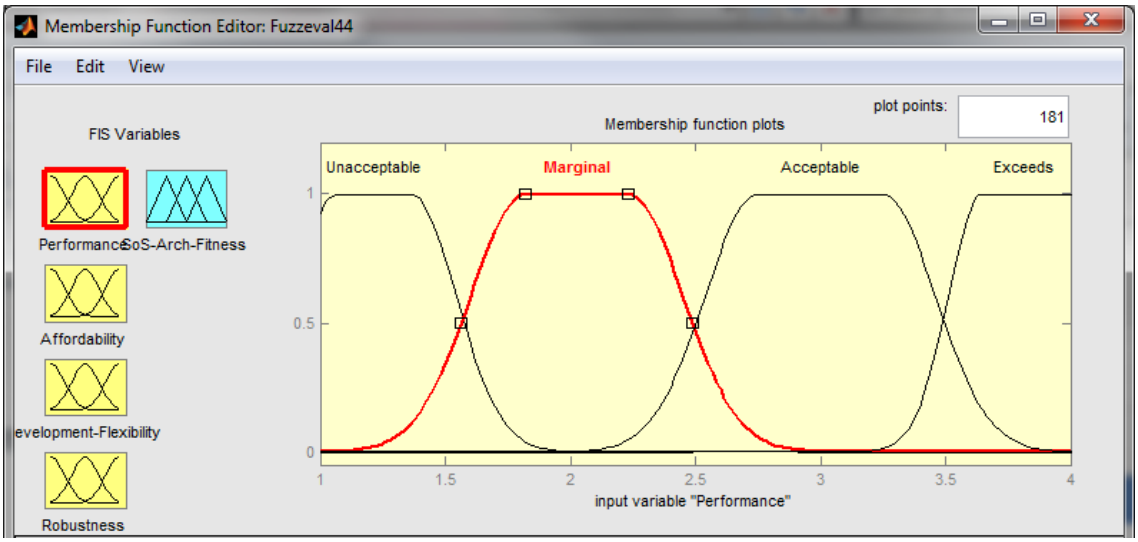


Figure 3.8. Matlab Fuzzy Toolbox display of typical membership function shapes

3.3.3 Mapping Attribute Measures to Fuzzy Variables. The generic membership function range is the 'universe of discourse.' This typical range, discussed in section 3.3.2, must be mapped to the real world values of the domain specific SoS.

The mapping can be done inside Matlab so that Figure 3.8 would be scaled in real units, but that requires working in a tedious GUI. It can also be done by mapping the key shape points of the scaled MF to the real world values. In a real problem, this mapping of ranges for each attribute would come from the problem definition and the stakeholders' beliefs and desires discovered during the model building step of the method. Examples could be estimated values for cost of the SoS, or performance in terms of square miles searched per hour, or tons of freight delivered per day in another type of problem. The probability of success, or the number of shipments, or other attributes would have desired thresholds that define the levels of performance in each attribute, such as: unacceptable, marginal, acceptable, or exceeds expectations in a four part granularity for each attribute. The judging criteria may take on a wide variety of terminology and of forms, depending on the domain. Any degree of granularity is possible. An even number of gradations were chosen in this instance to avoid the possibility of an evaluation question being answered in the middle. Odd numbers of gradations tend to allow stakeholders to answer too many valuation questions disproportionately in the middle during the interview process, while even numbers of gradations force the choices to be above or below average. This depends on one's problem and particular stakeholders, of course.

Figure 3.9. shows a typical mapping between real world values on the left, and the fuzzy variable on the bottom. Note that there is no requirement for the mappings to be linear. Figure 3.10 shows affordability and robustness mapped to their fuzzy values. All the attribute membership function values need to be a matched set, with a matched set of attribute models. In this case, robustness depends on the range of the values of performance. Therefore, if the maximum performance doubles due to a change in the

model, then the real world robustness map would need to change as well. The real world values for affordability are dollars, and robustness is the maximum loss of performance when removing any system, but they are mapped as negative values here, because less is better. This allows the fuzzy attributes to be plotted as monotonically increasing. Minor kinks in the mapping lines show that the slopes of the membership function maps do not need to be constant.

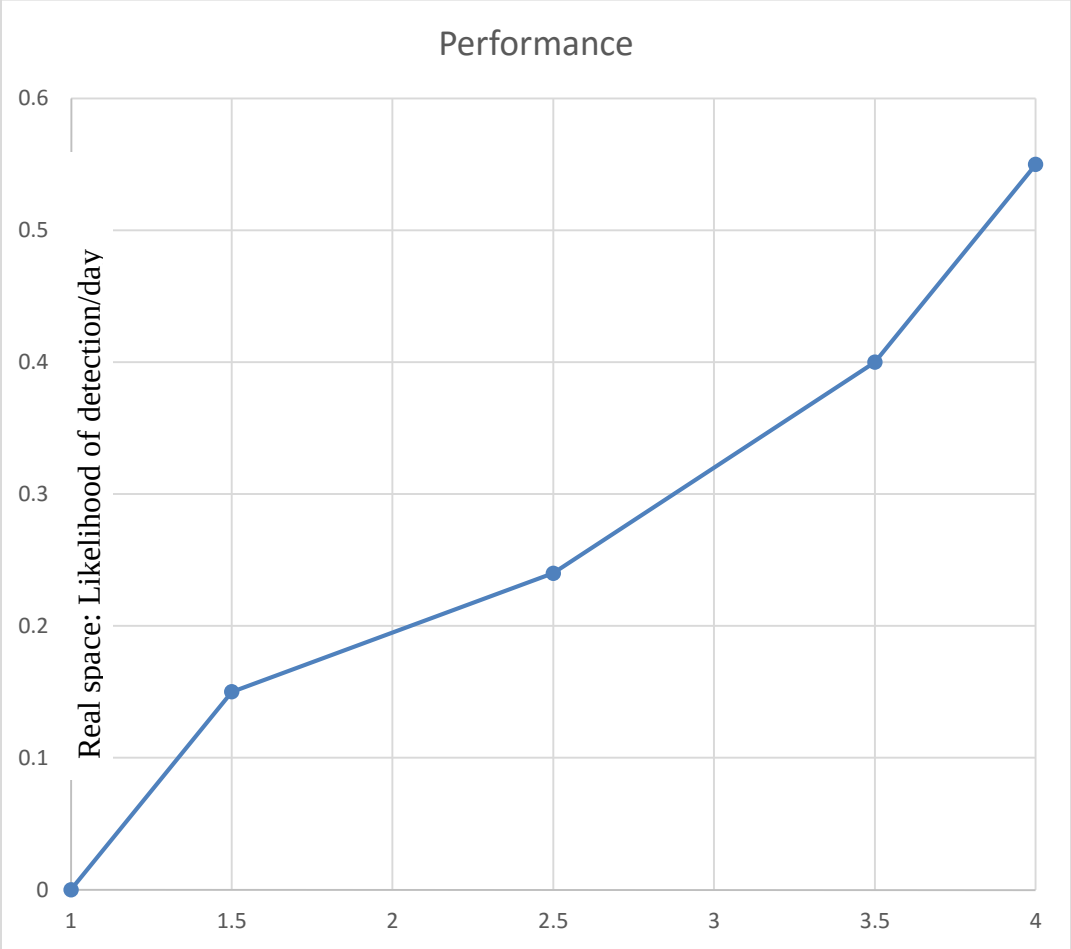


Figure 3.9. Map from fuzzy variable on horizontal axis to probability of detection on left

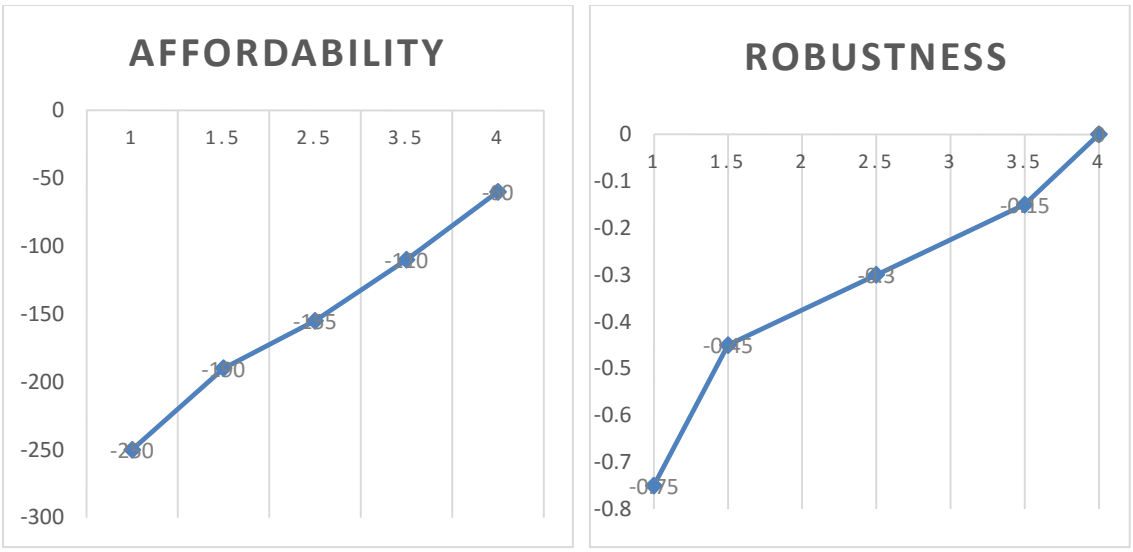


Figure 3.10. Attribute values, mapped to fuzzy variables

3.3.4 Exploring the Meta-Architecture Space to Set MF Crossing Values.

To explore the entire meta-architecture space for the demonstration of the method, a novel approach to defining the membership function sizes is used. After defining the draft attribute calculation algorithms to depend on the meta-architecture, a random selection of chromosomes with a wide variation in the number of ones is evaluated. The range of attribute values is examined to re-set the edge values of each fuzzy gradation of evaluations in each attribute to allow a solution. The MF edge values also need to be adjusted for attributes that depend on other attributes, such as robustness being dependent for absolute size on the performance range. With real world data, for a given set of rules, there is no guarantee that a solution is possible. In order to demonstrate the method, there must be achievable solutions. By exploring the values of each attribute over a range of chromosomes, the modeler may be able to find compromises that will work. This may require changes to the rules or desired values in the outcome. With real world problems,

another round of discussions with the stakeholders may be required to vet the model, rule, or MF definition changes required to make the solution method converge. This is called the value exploration phase of the model development. An example of this exploration approach is shown in Figure 3.11, with the explanation of the graphs in Table 3.4. These charts show that there are achievable solutions within this architecture model:

- A general trend toward more performance, robustness, and affordability with increasing numbers of interfaces
- The penalty peaks in the middle of the range
- There are several good starting points on the graphs.

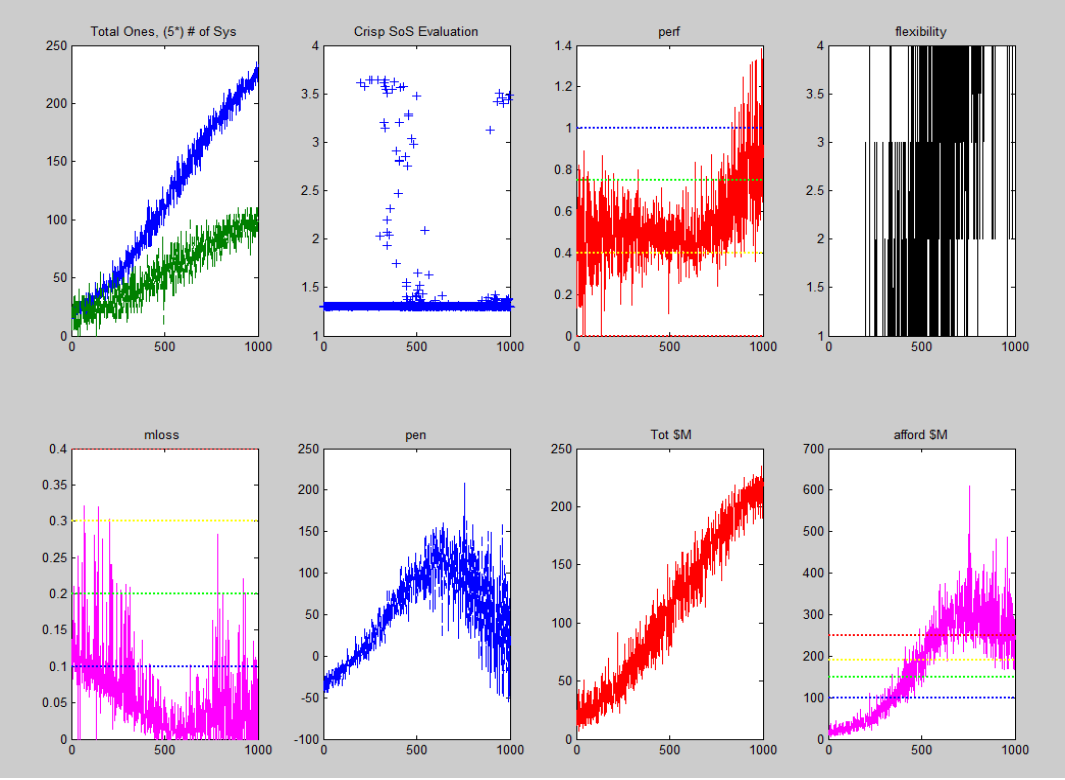


Figure 3.11. Setting the membership function edges for the attributes with value exploring

Table 3.4. Explanation of value exploring graph pages during early model efforts

1) On the first row of graphs, the number of ones in the whole chromosome (in blue) and five times the number of systems in the chromosome (in red) plotted together on the same scale
2) The overall SoS assessment on the 1 to 4 scale of unacceptable to exceeds expectations
3) The performance of each of the population chromosomes, with dashed lines of different colors representing the edges of the membership functions
4) The flexibility attribute evaluation of each chromosome
5) On the second row of graphs, the maximum loss in performance by successive individual system removal of each participating system – that is, the robustness attribute
6) The value of the penalty/reward function for using unachievable/achievable interfaces for each chromosome in the population
7) The total cost for each chromosome, and
8) The affordability attribute, which is the total cost modified by (one minus epsilon) raised to the penalty/reward power, as described in section 2.3 above.

By running a few thousand random chromosomes (with the biased total number of ones, but still randomly selecting systems and interfaces) through the fuzzy evaluation subroutine, one can settle on adequate values for the membership function edges to show there are good solutions possible within the model as shown in Figure 3.11. This is not yet the ‘finding the best chromosome’ part of the method, but only finding a set of membership function edges so that one can be sure of finding some acceptable chromosomes during the GA from which to select better mutations from each generation. One can also see similar shapes of the functions for each of the attributes and the penalty function. One important feature is that tiny changes in the chromosome can have wide swings in the values of each of the attributes. The search for a ‘good’ chromosome is

really that, a search for it – it is not obvious that there will be a single optimum from the model so far.

It takes only a few minutes to run 1000 ‘almost’ random chromosomes through the exploration phase. Several iterations on selecting the mapping values for the boundaries between membership functions may be required. If one selects values that are too tight, such as demanding a high performance, the robustness limits may need to be adjusted. When the membership function edges change, the input domain specific costs and performances, and the limits for the robustness function are selected so that there are at least some chromosomes that are performing well, the next step is to run the full fuzzy model through the GA for 60 to 100 generations, as discussed in section 3.6.

3.4 NEED FOR MULTI-OBJECTIVE OPTIMIZATION (MOO)

Since there are so many stakeholders in SoS, there might be dozens to hundreds of concerns that must be tracked, traded among, and optimized to create an acceptable SoS architecture. Using the proposed meta-architecture, the independent variable is the presence or absence of the system or the interface. The architecture may be changed only by either adding or subtracting a system or interface. System costs or other input characteristics may be changed, or the algorithms for modeling the attributes may be changed, but that is a secondary effect compared to changing a one to a zero in the meta-architecture. The SoS evaluation may be changed in highly non-linear and discontinuous ways by the change of a single bit in the architecture. For most of the analysis, the total number of one bits in the chromosome is used as a shorthand for the independent variable. One could pursue this sorting in other ways, such as number of systems, number of interfaces, or exactly which bits change, but it seemed the most real to have an

individual system or its interface be present or absent either during development (when an irrevocable decision is made for this epoch whether not to participate in the negotiation phase of SoS development) or independently again during employment due to the operational concerns mentioned above. Even if a system manager decides to participate during acquisition and development, on the day the system is needed by the SoS it may still be unavailable due to maintenance or being assigned to another mission, for example. This problem is more likely in acknowledged SoS composition, where the systems still have their continuing missions as individual systems or components of overlapping SoS missions competing for resources.

It is not feasible to try to find or construct Pareto dominant surfaces under these conditions, while holding 'other variables' constant. On the other hand, some variables, such as how much increase in performance might arise by increasing interfaces among component systems, may be analyzed in this way, but the desired performance (a measure of effectiveness) may also need to be adjusted for the model make sense. This is because the range of possible performances could change so much for small changes in NCO advantage epsilon discussed section 2.3. This again violates the 'all other things the same' assumption that one makes for describing a Pareto front. Finding the Pareto non-dominated solutions within a small region of the input space is difficult because it is hard to know what one means by 'within a small region' in the meta-architecture. Shall it be defined as being within a small Hamming distance: by changing the ones present within one or two rows and columns of each cell in the upper triangular matrix of interfaces within a chromosome? Alternatively, is it within a Hamming distance by allowing bits anywhere in the starting chromosome to change? If one of the bits being

changed represents a system, then whole rows of interfaces change from being achievable to unachievable or vice versa. If the bit represents a communication system, then many more interface bits may change from achievable to unachievable, or the reverse.

3.5 NON-LINEAR TRADES IN MULTIPLE OBJECTIVES OF SOS

Fuzzy logic can be used to fit highly nonlinear surfaces even with a relatively small rule base. The commonly cited problem of dimensionality for fuzzy logic systems in fitting arbitrarily large input sets (Gegov 2010) does not arise in this problem because the number of inputs are small – limited to the KPAs of the SoS design problem. The combination of membership function shapes and combining rules allows one to fit quite nonlinear surfaces in the several required dimensions of this problem. Furthermore, the input variables are generally monotonic, increasing in value from the fuzzy value of 'worst' to 'best.' All the membership functions used in this effort (input and output) have been scaled from 1 to 4 for simplicity of display in this document, but that scaling is purely arbitrary. The actual scaling is through linguistic variables discovered through the interactions of the facilitator, SMEs, and stakeholders. They are typically terms such as "very bad," "good," "excellent," etc. For most attributes, there is a further mapping of the linguistic terms, such as 'excellent affordability is a cost between $8M and $10M,' or 'acceptable affordability is cost between $10M and $12M.' If the attribute evaluation elements can be categorized in such fuzzy terms as this, then relatively simple rules for combining them can result in a straightforward overall SoS evaluation from the resultant fuzzy inference system or fuzzy rule based system. Figure 3.12 to Figure 3.14 show how quite complicated assessment shapes can be represented through the combination of MF shapes and the combining rules of the FIS. Figure 3.12 shows the SoS fitness surface

versus affordability and performance in the ISR example. Figure 3.13 shows the impact on the fitness surface of changing the membership function shapes; the left example is four triangular MFs with four rules; the right example is four trapezoidal MFs with 10 rules. Figure 3.14 illustrates a very different shape for the SoS assessment surface for the large training SoS validation problem, with seven MFs and 18 rules, showing that very complicated function shapes can be represented by the combination of FIS rules and MF shapes.

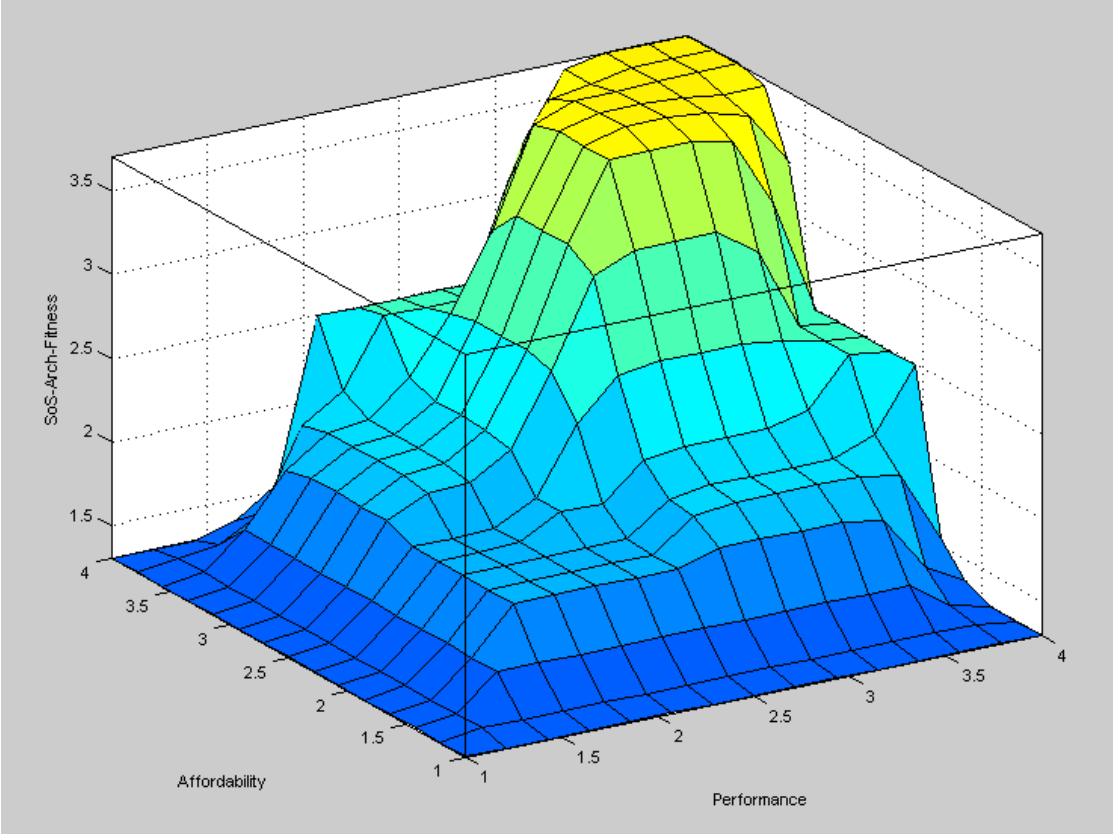


Figure 3.12. Nonlinear SoS fitness surface of the ISR fuzzy inference system (FIS)

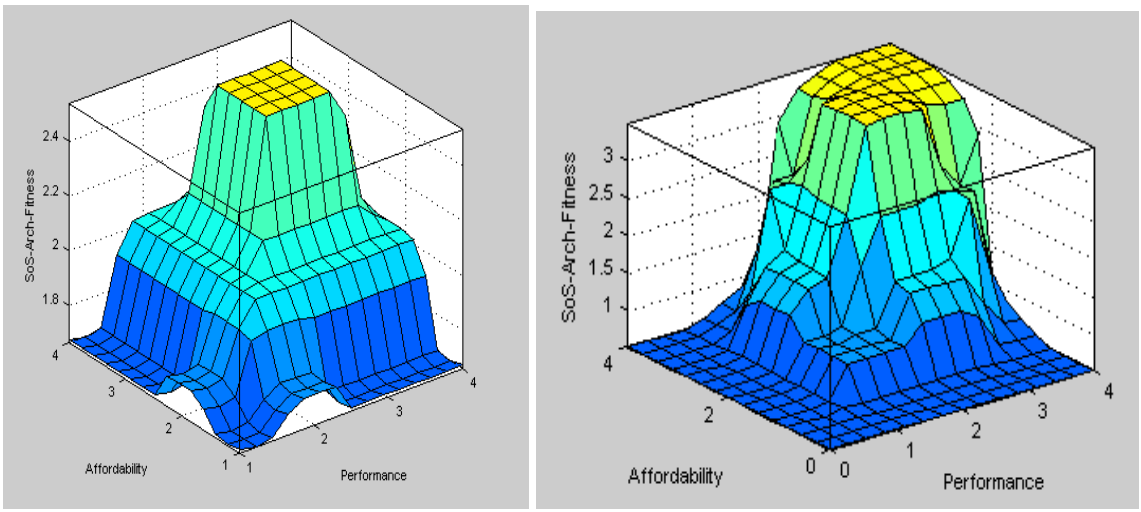


Figure 3.13. Alternate fitness shapes for different domain problems

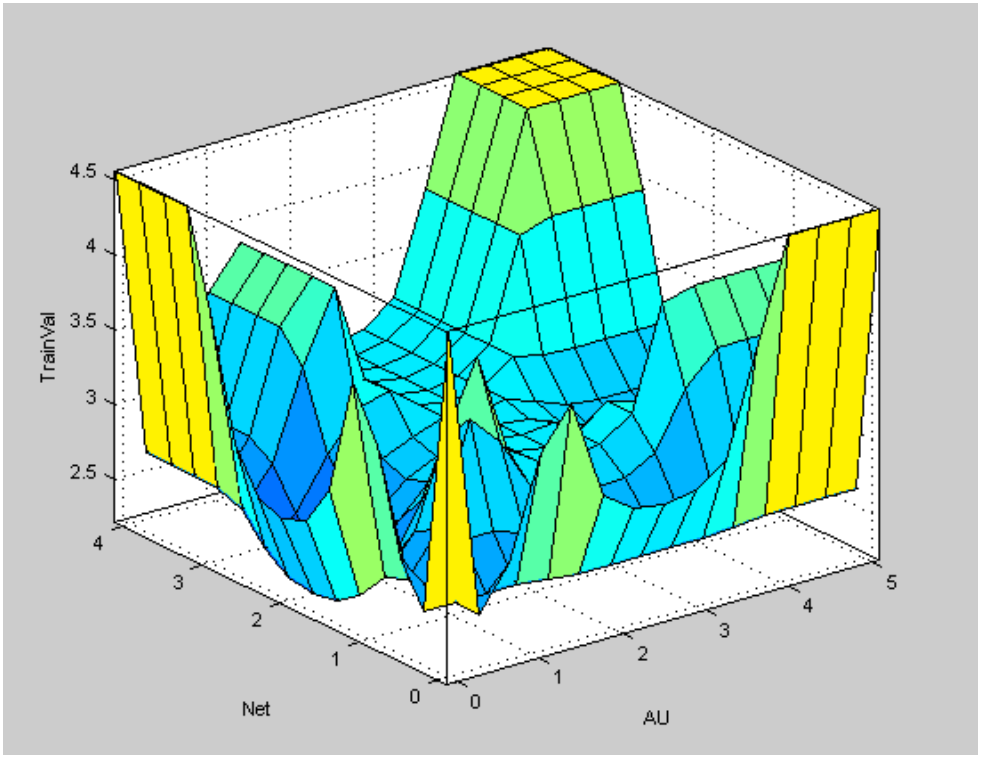


Figure 3.14. Fitness surface from the large training SoS validation problem

3.6 COMBINING SoS ATTRIBUTE VALUES INTO AN OVERALL SoS MEASURE

A Mamdani fuzzy inference system allows the combination of as many input attributes as desired (Fogel 2006). Each attribute is equivalent to an objective or dimension in a multi-objective optimization problem. Gegov expanded this concept to include networks of fuzzy systems, to cover deep and complicated problems with many dimensions (Gegov 2010), and uncertainties extending to Type II fuzzy sets. Nevertheless, if rules of the form discussed below (which are symmetrical), are combined with rules of the form 'if attribute one and attribute four are excellent, but attribute five is marginal, then the SoS is better-than-average,' etc., which allows for asymmetry or non-uniform weighting among attributes, then very complex evaluation criteria may be described for the SoS. Using membership function shapes other than those shown in Figure 3.8 also allows considerable tuning of the mapping of input attribute values (depending on the SoS architecture or chromosome structure in the model) to the output of the overall SoS quality or fitness.

A Mamdani Type I fuzzy rule set may also be called a Fuzzy Associative Memory (FAM) to combine the attribute values into the overall SoS fitness score. Attribute measures are converted to fuzzy variables from the mappings explained in section 3.3.3, the rules are followed to form a fuzzy measure for the SoS architecture (represented by a chromosome). That measure may be de-fuzzified back to a crisp value for final comparison in the GA through an equivalent mapping in the output space. The rules should be kept simple for two reasons: primarily it is easier for the analyst to understand and to explain them to the stakeholders, but also because a few rules within the fuzzy logic system can be very powerful in defining the shape of the resulting surface. Still,

some sensitivity analysis can be done on the rule sets, and results of minor changes in the rules may be displayed for comparison, all other things being kept the same. Rules are typically of the form: 'if all attributes are good, then the SoS is superb,' 'if all attributes except one are superb, then the SoS is still superb,' 'if any attribute is completely unacceptable, then the SoS is unacceptable.' A dozen or so of these rules can give an excellent estimate of the stakeholders' intentions, including significant nonlinearities and complexity (Gegov 2010). The Mamdani FIS does its best to satisfy contradictory rules simultaneously by simply including them both in the calculation of the resultant output value for optimization in the GA.

The linguistic form of some of these rules may be easier to express than the mathematical form. For example, 'if any attribute is unacceptable, then the SoS is unacceptable' can be expressed linguistically as a single sentence, but mathematically a separate rule for each attribute is tested alone to implicate the unacceptability of the SoS. If the rule can be expressed as a single sentence linguistically, as in Table 3.5, it will be

Table 3.5. Example of a few powerful Fuzzy Inference Rules for combining attribute values

Five Plain Language Rules
If ANY single attribute is Unacceptable, then the SoS is Unacceptable
If ALL of the attributes are Marginal, then the SoS is Unacceptable
If ALL the attributes are Acceptable, then the SoS is Exceeds
If (Performance AND Affordability) are Exceeds, but (Dev. Flexibility and Robustness) are Marginal, then the SoS is Acceptable
If ALL attributes EXCEPT ONE are Marginal, then the SoS is still Marginal

counted as only one rule. The rules come out of linguistic analysis of the stakeholder interviews, with some normative smoothing by the facilitator. At worst, if consensus cannot be reached on a rule statement among the stakeholders, a version of the analysis with the rule expressed both ways can be compared for sensitivity to that rule. This approach can also help explain the issue to the stakeholders.

3.7 EXPLORING THE SoS ARCHITECTURE SPACE WITH THE GENETIC ALGORITHM (GA) APPROACH

Having developed a method of evaluating architectures based on presence or absence of any combination of systems and interfaces within the meta-architecture, this evaluation may be used as the fitness measure for selection for propagation to a new generation within an evolutionary algorithm. One class of evolutionary algorithm is the genetic algorithm (GA). The key feature of a GA approach is to evaluate the overall fitness of a series of chromosomes in a 'population.' One then sorts the chromosomes by their fitness, and proceeds to a next generation through mutations, crossovers, or 'sexual reproduction' of a fraction of the better fitness chromosomes in that generation. Mutation rates, crossover points, special rules for certain sections of the chromosome (genes), or deciding which parents are combined, can all be varied as part of the GA approach.

The GA first generation starts with a population of random arrangements of chromosomes built from the meta-architecture, which spans the search space, then sorts them by fitness. A fraction of the better performing chromosomes is selected for propagation to the next generation through mutation and/or transposition. A few poorly performing chromosomes may also be included for the next generation, to avoid the

danger of becoming stuck on a purely local optimum, although proper selection of mutation and transposition processes can also help avoid this problem.

3.8 COMBINING THE FUZZY APPROACH WITH THE GA APPROACH

In order that the GA work with any string of bits within the meta architecture, the algorithms for evaluating each attribute must work for any string of bits. The results of individual attribute evaluations may take on a large range of values. When the desired and tradable values of the attributes, and the algorithms for evaluating them, are determined from the SoS stakeholder interviews, the range of values of each attribute is pre-determined. The entire range of possible values is the 'universe of discourse.' In each dimension or attribute, the entire range is mapped contiguously to the granularity described by the membership functions. There is no guarantee that any arrangement of systems and interfaces will be found to be acceptable. Because this effort was to develop and explore the method, and the example SoS were largely fictional, all the model parameters could be adjusted to find examples that would work. The key to this adjusting process was to plot the attribute evaluations against the number of ones in the chromosome. Figure 3.15 and Figure 3.16 show changing the MF edges for small, 25 chromosome population examples. The shapes of the attributes are similar, but the fuzzy value maps are adjusted.

Biasing the random number generator to produce a population of chromosomes with varying numbers of ones allowed an exploration of chromosomes from various regions of the meta-architecture. By iterating adjustments of the attribute membership function edges against a population of randomly generated (but biased in the number of

ones) initial populations of chromosomes, an acceptable picture of the SoS behavior could be determined.

When a few hundred chromosomes are present in the exploratory population, one can get a very good idea of the shape of the behavior of the meta-architecture space as a function of the number of interfaces between systems of the SoS, shown in Figure 3.17. More systems and interfaces generally leads to more of all the attributes: performance, flexibility, robustness, but to more cost as well (= less affordable). However, one can also see that the trends are noisy, and not perfectly correlated as shown in the ISR model in Figure 3.17. In the example on the left, too many good SoS are found because the MF edges are set too low. There is not enough discrimination in the combination of MF values, attribute evaluation algorithms and fuzzy inference system rules. On the right, performance, robustness and affordability MFs are mapped better; fewer SoS are in the exceeds range The exploration phase allows the setting of the MF edges to take advantage of the variability in the evaluations to drive the GA search toward regions that look more likely to produce a decent compromise from among the competing attributes.

One needs to be in a reachable region of the SoS attribute space, or the universe of discourse, defined by the MF edges of the fuzzy inference system when it is mapped back to the real world. It is of little value to have an architecture that produces $100M solutions when the only acceptable value is less than $50M. Therefore, some level setting of expectations, tuning of algorithms, and of the input domain data may all be necessary to reach a reasonable ‘space’ within which to attempt optimization with the GA. This is the function of the exploration phase of the process and includes going back

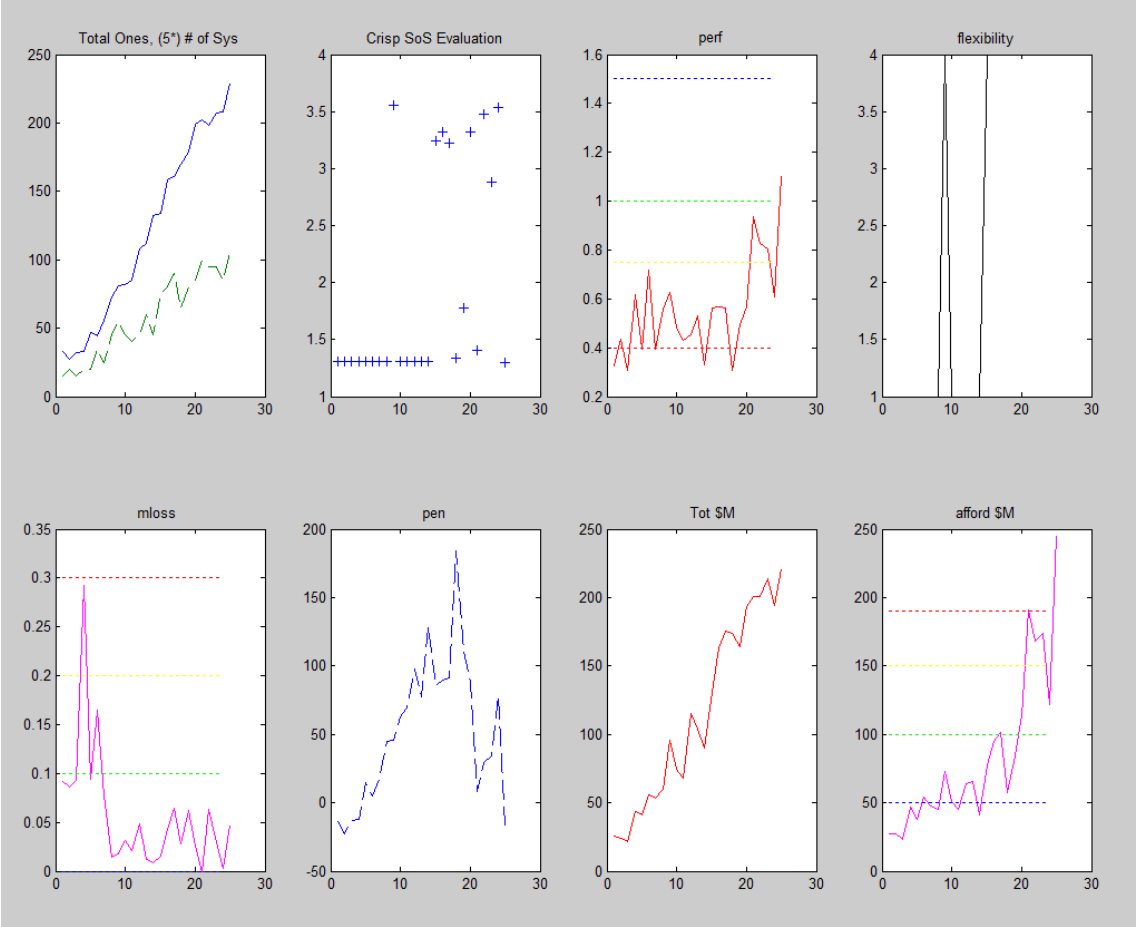


Figure 3.15. Exploring the meta-architecture - 25 chromosomes, 22 systems, Example 1

to the stakeholders to attempt to adjust their thinking when they have completely unrealistic expectations.

3.9 HEURISTICS

Heuristics may help find solutions more quickly, and the discovery of heuristics is important to finding better and/or faster solutions to many types of problems (Maier and Rechtin 2009). However, by definition, the reason a heuristic works is not strictly known (Blanchard and Fabrycky 2010). Heuristics may bias the discovered solution by

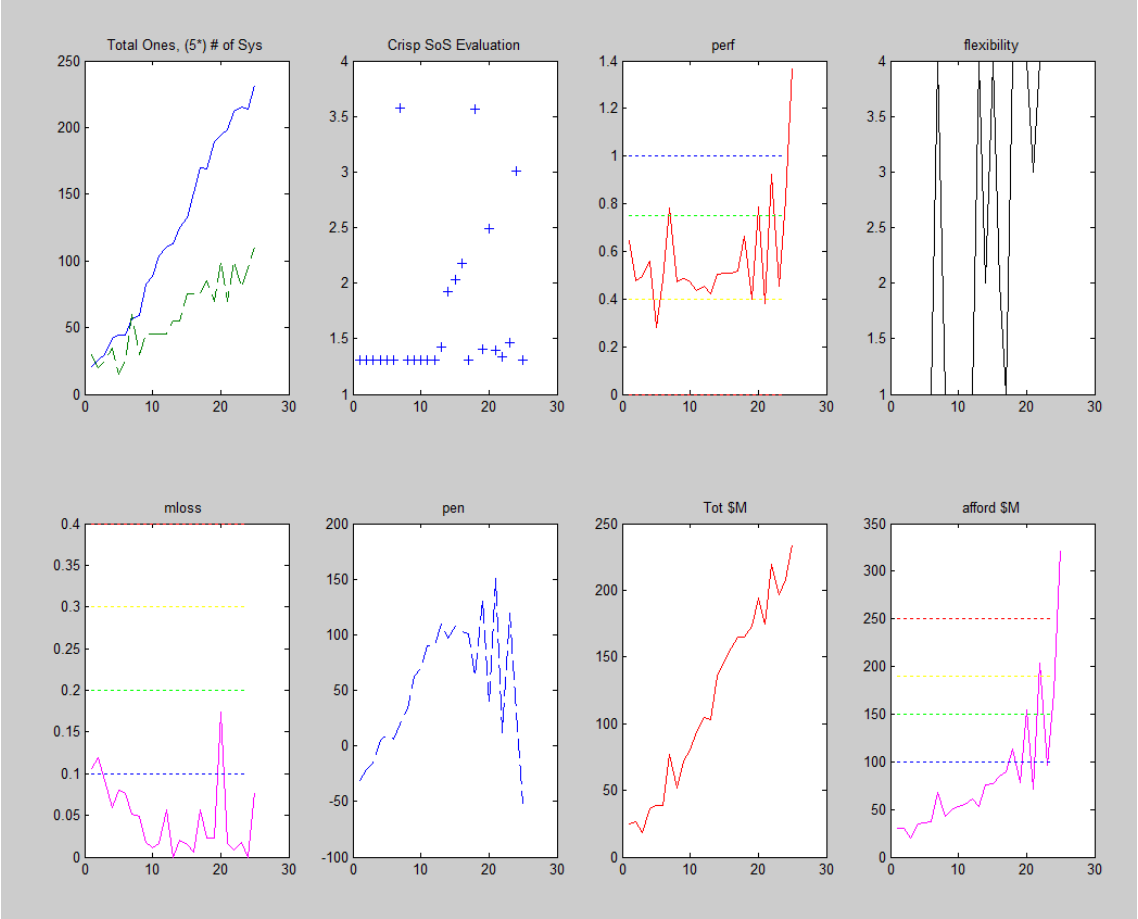


Figure 3.16. Exploring the meta-architecture to map membership function edges, Example 2

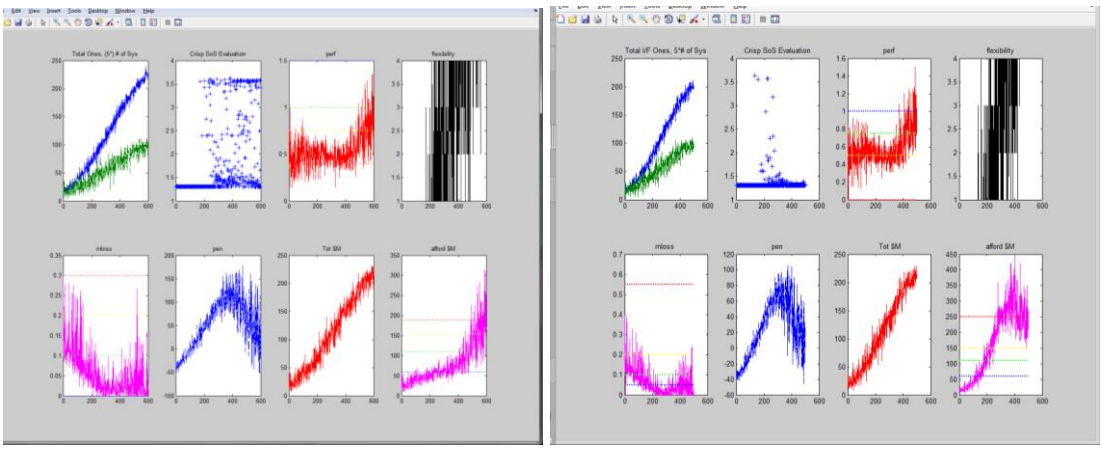


Figure 3.17. Exploring biased, but still random populations to set the membership function edges

discarding possibilities in unknown ways. Even though many heuristics are known to be biased, they are used both intentionally and unconsciously (Taleb 2004). There are no guarantees that any particular heuristic will continue to be useful (as it has been in the past) on a new problem. Heuristics are common sense derivations from experience in solving similar problems, but if the reason they worked was fully understood, they would be part of the formal solution method and not classed as an heuristic. The methods of solution worked out here attempt to avoid heuristics because we do not yet understand the nature of a 'good' SoS solution well enough to trust any heuristics. The example problems are not so large as to require extensive use of heuristics to reach a reasonable solution in quite reasonable times, either, which is a standard reason for relying on an heuristic to narrow the search space and reduce the time in computing a solution (Blanchard and Fabrycky 2010).

3.10 DISPLAYING THE RESULTS OF COMPLEX SoS ANALYSES

A key feature of understanding problems of this nature is to be able to visualize the solution. While the architecture framework was easy to describe in text, and even to draw pictures of what was meant, until the upper triangular visualization was discovered, it was difficult to see patterns or to compare two solutions in a meaningful or easy to use manner. Figure 3.18 shows the format of a chromosome, color coded to show used/unused systems and interfaces as colored versus the dark brown color for unused. The red and green colors show where ones exist; Green for an achievable and used interface or system. Red is for attempting to use an unachievable interface, and blue is for an unused interface that would have been achievable, if it were used. Figure 3.19 shows this display for the 29 system SAR SoS. It is not automatically true that the

overall fitness of a chromosome would be enhanced if the blue interfaces were used. It costs money and time to develop interfaces (normally), so the cost could go up if they were used. It is difficult to predict how the other attributes would be affected by using the blue (achievable but unused), or deselecting the red (unachievable but selected) interfaces. Another reason not to discard selected interfaces arbitrarily is that the model is intended to be used to mimic the wave model evolution of the SoS over several epochs, when new systems might be persuaded to join, or longer term modifications come to fruition, and previously unachievable interfaces now switch to achievable ones.

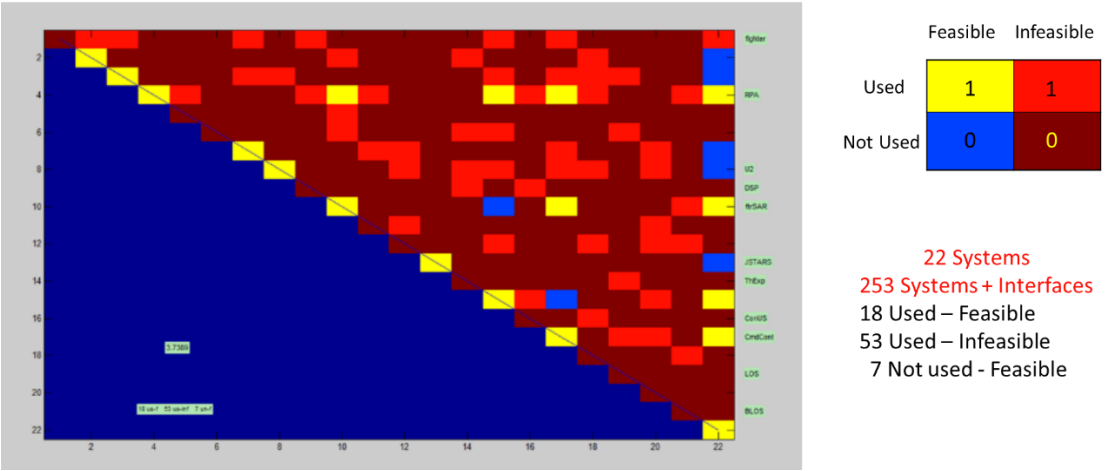


Figure 3.18. Upper triangular form of chromosome, with color codes for used and achievable (or feasible) interfaces

The four representations in Figure 3.20 are equivalent ways for showing identical participating systems and interfaces in an SoS. The upper triangular matrix on the upper

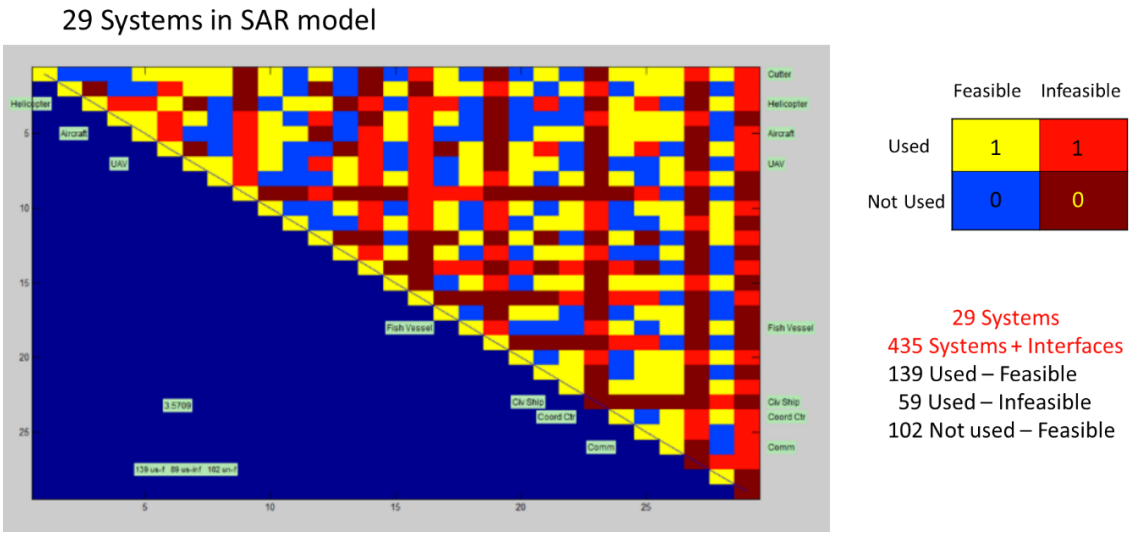


Figure 3.19. Color coded achievable/unachievable interfaces for a SAR SoS

left also shows the achievability/unachievability of the interfaces through color-coding. The 'ojo de dios' display, sometimes called the 'circle' display, in the upper right shows the systems' presence by the number at a vertex, while the interfaces are shown by the connecting lines between the vertices. The triangular matrix at the lower right shows only the presence of the systems or interfaces through the color coding, ignoring the achievability. Finally, the linear representation at the bottom shows the highly compressed systems and interfaces presence by the color coded downward pointing 'teeth' where there is a one. The alternating color bands along the top show the systems on the far left and the interfaces of each system in the same order as the rows of the triangular matrices. The triangular matrix representation is far superior for identifying the position of the interfaces (a key element of defining the architecture) when the number of systems becomes large.

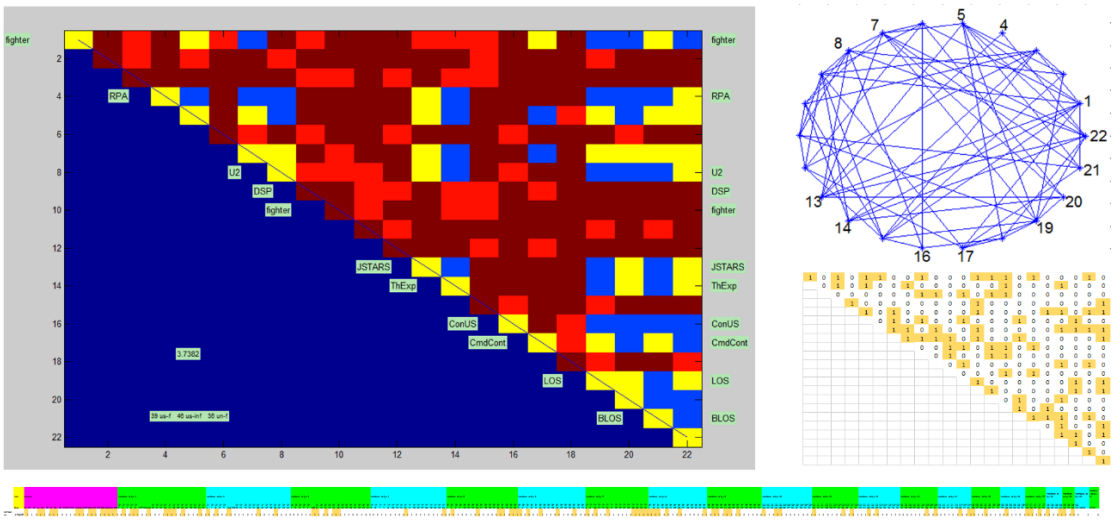


Figure 3.20. Four equivalent methods of showing the systems and interfaces in an SoS

3.11 MODULARIZING THE METHOD

Each component of the FILA-SoS approach, and each attribute model was designed to be modular, so that if the definition of performance or other evaluation factors, such as cost, time to deliver, etc. changes due to new information or the development of an improved algorithm, the other components do not need to be changed. If it seems that reasonable results are produced through the process from simple models, model parts may be replaced with more accurate models, or models validated by a standards agency. The combined model, with its input data, algorithms for combining system capabilities to SoS capabilities, evaluation criteria, and GA tuning factors must be independently validated, then tested together to insure that the whole process produces reasonable SoS architectures.

This part of the FILA-SoS effort produces architectures to be handed off via a well-defined Excel spreadsheet interface to the negotiating team of agent-based models to achieve a realized SoS each epoch. Those models test various negotiation strategies and

policy incentives in the creation of the final SoS from the suggested component system/interface architectures produced here.

3.12 VALIDATING THE DEVELOPED MODEL

Validation is the agreement from the customer that the system (or SoS) does in fact provide a solution to the problem it is intended to solve. Verification is the furnishing of proof that the designed system is what was produced. If systems engineering was done correctly throughout the program, customer involved design reviews at several stages should have validated that the design should produce a system that will satisfy the customer. However, with the type of acknowledged SoS in discussion here, already produced (legacy – sometimes long out of production systems that are in the sustainment phase of their life cycle) are being slightly modified (if at all) to meet the new need. Much of the normal life cycle validation process for a system development has been skipped over when the SoS is composed mostly from legacy systems. Validating the FILA-SoS component models is accomplished by a series of steps from the very beginning of the concept development through the 'end' of the SoS design process. The 'end' of the process is really only the start of the next wave in the wave process of SoS evolution, when the process starts over, possibly with minor changes in the environment, goals, or component systems. The model validation steps include:

- The first validation step is that domain SMEs must help write the original goal statement for the SoS. They use the appropriate vocabulary to begin the concept development, and begin the documentation process with equivalent descriptions to the DoDAF all-view viewpoints

- The fact that this method is intended for an acknowledged SoS means that the component systems are persuaded, not directed to join. They must 'buy in' to the SoS concept and their part in it, the same way that all the other stakeholders do. Only one other element is more important to validation
- The management staff of the SoS must be open to suggestions, questions, and issues being raised by the prospective and committed participating systems' personnel to the purpose, goals, plans, integration methods, evaluation algorithms, or proposed testing for how they will contribute to the SoS. This is the only way that an acknowledged SoS, with peer component systems, can work.

One of the keys to achieving validation is for the SoS management to be 'honest brokers' of information, that is, actively seeking constructive criticisms and suggestions, and following up on action items from all interactions Regular reviews with the community of SoS participants aid the following goals:

- To judge progress
- Encourage completion of development and integration of the interfaces demanded by the choice to participate
- To adjust and socialize (i.e., get consensus on) plans, whether things are better or worse than the last accepted joint SoS plan.

The best way to validate the SoS modeling effort requires:

- Openly sharing information with all the stakeholders
- Actively asking for inputs, suggestions, and criticisms
- Making it a collaborative effort
- Getting everyone to agree, or at least not object.

In short, it requires an open culture of using the ‘exploration’ analysis phase of the SoS architecting process to socialize what participation, combined with domain data and attribute definitions, together mean to an overall SoS quality result. This is what is called for in current SE standards (ISO/IEC/IEEE 2011) (ISO/IEC/IEEE 2008).

3.13 HOW TO KNOW WHEN ONE HAS A GOOD SOLUTION

There are several ways to check on the validity of solutions from the fuzzy GA. The first step is to examine the selected chromosome to determine if it makes sense on its face. This consists of at least the following steps:

- Check the evaluation of individual attributes to ensure the model algorithms seem to be working properly
- Check that the fuzzy inference system rules are being properly applied
- Make a few conscious mutations in the solution chromosome to see if either the KPA evaluations or the SoS assessment can be improved
- Socialize the solution among stakeholders and SMEs to find out if they agree that it is a good solution

The validity of the process may be checked by the following steps:

- Alter some of the input data, such as operations costs, or performance values, and see if the new solution seems to take those changes into account properly
- Alter the membership function edge mapping to see if those changes move the solutions in an appropriate direction
- Change the relative value of reward and penalty for achievable/unachievable interfaces

- Membership function basic shapes may be changed between trapezoidal, Gaussian rounded trapezoidal, triangular, and Gaussian.

All these validity checks were performed numerous times on the all the example solutions from the fuzzy GA.

3.14 USING THE SoS ASSESSMENT WITH NEGOTIATION MODELS

In addition to the architecture definition itself, budget, schedule, and performance functions are assigned to the individual systems. The chromosome tells each system what interfaces to develop. The performance, budgets and schedules in the input data are the SoS manager's best estimate with limited knowledge. It is assumed the systems may know better what performance they can deliver within the proposed funding and schedule. Therefore, the systems negotiate with the SoS manager to update the existing cost, performance, or other attribute estimates. The negotiation model assumes the individual systems do not share information with other systems during negotiations. Individual systems may be negotiating for funds to create an interface with another system, while the other system may be refusing to participate in this epoch. It is another simplification to not allow systems to share information during negotiations, but not that far removed from reality, either. System modification possibilities and funding are frequently closely held information, or even classified, so that normally the systems do not freely share that information among themselves. The negotiations attempt to achieve the GA proposed SoS architecture. Sometimes the systems decide they cannot agree to the proposed funding for a performance commitment, and drop out, or become non-participants. Sometimes they decide they can actually deliver a little more performance than was requested, or for less funding.

If updates are made to the systems' cost, performance and schedule inputs during negotiations, those should be fed back to the evaluation inputs. At first order, one can simply rerun the original evaluation model with the negotiated systems and interfaces, because any negotiated changes are generally small changes to the initial estimates and any particular system's data forms only a small contribution to the answer.

The next chapter will show how the method was applied to the selection of an architecture for several interesting SoS of different styles and sizes to create the input domain data files. Several outputs are demonstrated, with a discussion of sensitivity analysis to input data variations.

4. APPLICATION OF THE METHOD

4.1 DOMAIN DATA GATHERING

The method developed in Chapter 3 was originally employed on an intelligence, surveillance and reconnaissance (ISR) example inspired by history. The SoS attributes, their definitions, ranges of values used for membership functions and their definitions, and some of the evaluation algorithms were developed over a year in weekly meetings of a subject matter expert (SME) group. The group included academics, military members, and SoS SMEs from government. The OOTW example of section 4.1.2 was created to test the method on a similar size but slightly larger SoS, with different capabilities but differently purposed and differently performing group of systems and evaluation algorithms. The fuzzy assessor for OOTW was the same as ISR with the exception of adjusted membership function edges. The SAR example in section 0 was selected to show the method and code worked on SoS with different number of systems and capabilities. Completely new attribute evaluation algorithms were used, even though the same attributes were used in the fuzzy assessor. Two fuzzy assessors were used on SAR, one still using trapezoidal membership functions, the other with triangular membership functions.

The MITRE ‘toy’ problem was used because it had been studied previously. The original toy problem is only five systems, and all are used all the time. This did not fit the FILA-SoS paradigm, so MS&T researchers created a 22 system toy problem, with multiples of each type of the five systems. Another MITRE suggested very large validation problem of a live, virtual, constructive training SoS is described in section 4.1.5. The method of Chapter 3 was applied to a DoDAF description of the architecture,

arriving at a model with 111 systems and 74 capabilities. Seven attributes with five membership functions were defined for this problem. Section 4.1.6 discusses how the method could be applied to the extremely large problem of global air traffic management.

4.1.1 Historical Example – Gulf War ISR Domain Model. A guiding physical example is taken from relatively recent history. During the 1991 Gulf War, Iraqi forces used mobile SCUD missile launchers called Transporter Erector Launchers (TELS) to strike at Israel and Coalition forces with ballistic missiles. Approximately 50-60 TELs were hidden in the western Iraqi desert, from which Iraqi forces launched somewhere between 100 – 200 missiles during the 43-days of intense combat. The Iraqi forces had developed new techniques called 'shoot and scoot' that allowed them to reduce the TEL vulnerability time to half an hour. This included the time to come out of hiding, set up, launch, and return to their hiding places. This was only one third of pre-war intelligence estimates of 90 minutes, and a great surprise to Coalition planners (Thompson 2002). While the relatively inaccurate Scuds were not a tactically significant factor in the war, their potential for carrying chemical or biological warheads meant that they had a significant strategic impact on morale and cohesiveness of the Coalition. Israel had been persuaded to stay out of the conflict, but that decision was threatened by Scud attacks on their cities. The Coalition included many Arab countries, who threatened to withdraw if Israel joined the conflict. It was, in fact, a very successful tactic for the Iraqi forces, deflecting significant combat and diplomatic power from the central purpose of the Coalition. Therefore, the TELs became a "high value, fleeting" target for Coalition forces (Rostker 2000).

Existing intelligence, surveillance, and reconnaissance (ISR) assets and processes at that time were inadequate to find the TELs during their shortened setup and knock down time of visibility. The 'uninhabited and flat' terrain brought to mind by the term 'western desert' was in fact neither of those things, with a significant population of Bedouin herders and their families, significant traffic (100,000 vehicles), and thousands of wadis with culverts and bridges in which to conceal the TELs and obscure their movement. In addition, the Iraqi forces produced some very fine camouflage and realistic decoys, again surprising Coalition planners (Rosenau 1991). Even though several thousand sorties were flown against hundreds of TEL firing opportunities, TELs were spotted only 11 times, and the contacts were lost before completing an attack eight of those 11 times. The average time between spotting and arriving at a potential target with a strike aircraft was about 90 minutes, which might have been marginally acceptable before development of the shoot and scoot tactic (Thompson 2002). This offers a clear example of existing systems being inadequate to address a highly important mission. Potentially, some relatively low cost, quick changes, and the joining together of existing systems might have been able to create an SoS capability to perform the mission better.

Applying the method described in Chapter 3 above to a slightly fictionalized version of the Gulf War ISR problem with a small team of subject matter experts (SMEs) resulted in the following hypothetical input domain parameters for treating this as an SoS problem. The characteristics of the SoS reached by consensus of stakeholders and SMEs are listed in Table 4.1. Most of the suggested important requirements of the ISR SoS was distilled down through the SME discussions to the following four attributes, measurable by operations on the chromosome describing the SoS:

- Performance is simplified to the sum of the square miles of terrain able to be searched by the SoS divided by the total search area; equivalent to targets found per day. A marginally good performance for reasonable SoS would be a probability of finding and destroying a single TEL per day. This is far better than the actual performance during the war. An original performance model was developed in great detail, but the details were regarded as too arcane for most reviewers. The original performance model is detailed in Appendix A as an example of a reasonably sufficient operational performance model.
- Affordability depends on the sum of the total cost ranges of development and operation of the SoS; less cost is more affordable. Occasionally affordability had the inverse of the netcentric boost applied to it, to make it a little more nonlinear.
- Flexibility in terms of development – multiple sources (systems) are available for each required capability contributed to the SoS; less sources means less flexibility.
- Robustness, defined as the smallest maximum loss of performance by successive removal of each participating system (Pape and Dagli 2013) (Deb and Gupta 2006)

Performance and affordability are adjusted by a netcentric factor in the exampled to keep them from being too linear, depending on the interconnectedness (number of interfaces), and proper use of achievable interfaces, as represented in the chromosome.

Table 4.1. ISR SoS domain example characteristics

Overarching Purpose of SoS	ISR & Targeting of Gulf War Iraqi Scud Missile TELs
Unique value of SoS	Existing non-networked systems not doing the job
SoS Measures of Effectiveness	Probability of successful engagement per day
Issues that might limit effectiveness	SCUD TEL concealment and countermeasures Short time of exposure of TEL before and after launch
SoS features that might greatly increase effectiveness	Improved probability of detection in presence of concealment Significantly Improved speed of response
Desired Effectiveness	About 1 successful engagement per day or more
Stakeholders	Operating commands, system operators/crew/maintainers, intel agencies, coalition partners, regional states, system program offices, troops in theater, contractors, Congress, DoD, enemy forces
ROM Budget: Development	About $40 Million
ROM Budget: Operations	About $40 Million
Attributes of the SoS, and range limits for fuzzy evaluation	Performance – from about 0.5 to 1.0 successful targetings per day Affordability – a few dozens of millions of dollars Robustness – less than 15% loss of capability for loss of one system Flexibility – prefer no single sources for component capabilities
Capabilities of contributing systems	EO/IR Synthetic Aperture Radar Exploitation Command & Control Communications

The capabilities of the ISR SoS, contributed by the component systems, were broken down into the following five elements:

- Electro-Optic/InfraRed (EO/IR) search capability
- Side looking, synthetic aperture radar (SAR)
- Command and control facilities

- Exploitation centers (smaller ones in theater and a large one in the continental US. (CONUS))
- Communication capabilities, both line of sight (LOS) limited to in-theater, and beyond line of sight (BLOS) for reachback to CONUS

Taking some poetic license with respect to the historical example, the following are the proposed types of systems within this SoS, with the non-communication systems limited to one primary capability plus communications.

- Fighters, some equipped with an EO/IR capability, some with SAR
- Remotely Piloted Aircraft **(**RPA**)**, equipped with better EO/IR capability
- U-2 aircraft, primarily equipped with EO/IR capabilities, but limited to film, so that system is not timely, but can help reduce the overall search area for the other systems, if it participates
- Defense Support Program **(**DSP**)** satellite system, that can surveil the entire area, but only provide notice on actual launch, reducing the time for the fighters to arrive before the TELs are hidden again
- JSTARS, with large SAR
- Control Stations for the RPAs or Air Operations Center (AOC)
- ISR data Exploitation and fusion centers
- Communication systems, LOS and BLOS, that enable the interaction between systems that make the SoS work.

A possible set of capabilities and costs of systems and interfaces for an SoS to address the Gulf War TEL problem are shown in Table 4.2. This resulted in an ISR SoS model with 22 potential systems of nine types, with five different capabilities among

them, with at most two capabilities per system. Later examples had more capabilities per platform, with more complicated performance models, but the ISR SoS model allowed a reasonable level of complexity to start.

Table 4.2. Domain model data for SoS with 22 Systems: Capabilities, Costs, and Schedules

System	Type Sub-System	Capability Number	Coverage sq mi/hr;	Develop $M/ epoch/ interface	Operate $K/hr per system	Time to Develop, Epochs	Number possible in SoS	System Number
Fighter	EO/IR	1	500	0.2	10	1	3	1-3
RPA	EO/IR	1	2000	2	2	1	4	4-7
U-2	EO/IR	1	50000	0	15	0	1	8
DSP	IR	1	100000* .01	1	1	1	1	9
Fighter	Radar	2	3000	0.7	10	1	3	10-12
JSTARS	Radar	2	10000	0.1	18	1	1	13
Theatre	Exploit	4	5000	2	10	1	2	14-15
CONUS	Exploit	4	25000	0.2	0	0	1	16
Control Station/ AOC	Cmd & Control	5	1	1	2	1	2	17-18
LOS Link	Comm	3	1	0.2	0	1	2	19-20
BLOS Link	Comm	3	1	0.5	3	1	2	21-22

The inputs from Table 4.2 were adjusted slightly to simplify the model by scaling all the capability contributions to be relative to square miles searched per hour. This allowed a simplified performance algorithm to be implemented in the fuzzy fitness assessor. The equivalent input data from Table 4.2 are shown in the Excel input sheet shown below in Figure 4.1. The modularity suggested in section 3.11 allows higher

fidelity models for either capabilities or attributes to be substituted relatively easily if they are available, after demonstrating the approach is viable with simpler models as used here. See Appendix A for a representative more detailed performance model. Table 4.3 shows how the membership function significant points were entered in the Excel spreadsheet of input data. Table 4.4 matches the input data to the mathematical explanations in Table 3.1.

Name	ISR										
NumSys	22	m									
NumCap	5	n	*sys has capability, costs, perf, deadline*				*1*	*2*	*3*	*4*	*5*
SysNo	**Type**	**Capability**	**I/FDevCos**	**OpsCost/I**	**Perf**	**DevTime**	*EO/IR*	*SAR*	*Exploit*	*C2*	*Comm*
1	fighter	1	0.2	10	10	1	x				x
2	fighter	1	0.2	10	10	1	x				x
3	fighter	1	0.2	10	10	1	x				x
4	RPA	1	0.4	2	10	1	x				x
5	RPA	1	0.4	2	10	1	x				x
6	RPA	1	0.4	2	10	1	x				x
7	RPA	1	0.4	2	10	1	x				x
8	U2	1	0	15	3	0	x				
9	DSP	1	1	0.1	8	1	x				
10	ftrSAR	2	0.7	10	15	1		x			x
11	ftrSAR	2	0.7	10	15	1		x			x
12	ftrSAR	2	0.7	10	15	1		x			x
13	JSTARS	2	0.1	18	40	1		x			x
14	ThExp	3	2	10	10	1			x		x
15	ThExp	3	2	10	10	1			x		x
16	ConUS	3	0.2	0.1	15	0			x		x
17	CmdCont	4	1	2	12	1				x	x
18	CmdCont	4	1	2	12	1				x	x
19	LOS	5	0.2	0.1	10	1					x
20	LOS	5	0.2	0.1	10	1					x
21	BLOS	5	0.5	3	10	1					x
22	BLOS	5	0.5	3	10	1					x

Figure 4.1. ISR domain specific input data

Table 4.3. Trapezoidal Membership Function crossover values

Lower Bound Attributes	1 Unacceptable	1.5 Marginal	2.5 Acceptable	3.5 Exceeds	4 (upper)
Performance	0.4	0.75	1.5	2	5
Affordability	-200	-100	-85	-65	-40
Flexibility	1	1.5	2.5	3.5	4
Robustness	-0.9	-0.6	-0.4	-0.2	-0.05

Table 4.4. Mathematical definition of variables for ISR domain example

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for ISR Model**
Name of SoS:	sos	1	ISR
Number of potential systems:	m	2	22
Number of types of systems:	t	3	11
Names of system types:	$sys\_typ_i : i \in \{1,...t\}$	4	sys_typ1 = fighter sys_typ2 = RPA sys_typ3 = U2 sys_typ4 = DSP sys_typ5 = ftrSAR sys_typ6 = JSTARS sys_typ7 = ThExp sys_typ8 = CONUS sys_typ9 = CmdCont sys_typ10 = LOS sys_typ11 = BLOS
Number of component capabilities:	n	5	5
Names of component capabilities:	$sys\_cap_i : i \in \{1,...n\}$	6	sys_cap1 = EO/IR sys_cap2 = SAR sys_cap3 = Exploitation sys_cap4 = Cmd & Control sys_cap5 = Communication
Binary meta-architecture upper triangular matrix:	$A_{ij} : i \in \{1,\ldots m\}, \ j \in \{i,\ldots m\}$	7	Selection of systems and interfaces between them
Individual systems of the SoS	$A_{ij} : i \in \{1,\ldots m\}, \ j = i$, also sometimes written as A_{ii} , or simply A_i	8	Numbered systems up to m=22
Achievable interface	$A_{ij} : i \in \{1,\ldots m\}, \ j > i$, and $A_{jk} = 1$, $A_{ik} = 1$, $A_{ii} = 1$, $A_{jj} = 1$, $A_{kk} = 1$, where A_{kk} is any communications system	9	Depends on both system interfaces with joint communications systems, and systems' presence in the architecture

Table 4.4. Mathematical definition of variables for ISR domain example (cont.)

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for ISR Model**
SoS main capability:	C	10	Detection of TELs
SoS performance in its large capability:	P_{SoS}	11	Expressed as probability per day of finding a TEL $\frac{(\sum_i^m a_{ii} P_i \sum_k^n c_{ik})}{\sum_i^m P_i \sum_k^n c_{ik}} (1 - \epsilon)^{Penalty}$
Component capabilities of systems:	c_{ij} : i ∈ {1,…n}, j ∈ {1,…m} (binary)	12	Whether each system posseses each capability
Performance of a particular system in its key capability:	P_i^{Ss} : i ∈ {1,…m}	13	Depends on the system; simplified down to a single gestalt number for this example; shown in Figure 4.1
Estimated funding to add an interface to an individual system:	FIF_i^{Ss} : i ∈ {1,…m}	14	$\sum_{i=1}^{m} (\sum_{j=1}^{i-1} a_{ij} * FIF_j^{Ss} + \sum_{j=i+1}^{m} a_{ji} * FIF_j^{Ss})$
Deadline for developing new interface(s) on a system:	D_i^{Ss} : i ∈ {1,…m}	15	Shown in Figure 4.1
Estimated funding for operation of all the participating systems during an SoS operation:	FOP_i^{Ss} : i ∈ {1,…m}	16	$\sum_{i=1}^{m} a_i * FOP_i^{Ss}$
Function describing the advantage of close collaboration within an SoS as a function of participating systems and interfaces:	$\mathcal{F}(A_{ii}, A_{ij, j\neq i},)$: i ∈ {1,…m}, j ∈ {i,…m}	17	$\sum P_{Systems} * (1 + \epsilon)^{(\sum Achiev.\ I/F - \sum Unachiev.\ I/F)}$

Table 4.4. Mathematical definition of variables for ISR domain example (cont.)

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for ISR Model**
Function for combining system capabilities into SoS capability C:	$C = \sum_k^n \sum_i^m A_{ii}\, c_{ki}$	18	$\frac{(\sum_i^m a_{ii} P_i \sum_k^n c_{ik})}{\sum_i^m P_i \sum_k^n c_{ik}} (1 - \epsilon)^{Penalty}$ Penalty = $(\sum Unachiev.i/f - \sum Achiev.i/f)$
Number of individual attributes the stakeholders want to evaluate the SoS over:	g	19	4
Attribute names to evaluate SoS architectures against (e.g., cost, performance, flexibility):	Att_k : k ϵ {1,...g}	20	Performance Affordability Flexibility Robustness
Number of gradations of each Attribute that become Fuzzy Membership Functions (MF):	h_k : k ϵ {1,...g}	21	4 Each
Fuzzy membership function names within each attribute (granulation = a, attribute = b):	MF_{ab} a ϵ {1,…h_k}, b ϵ {1,…g}	22	a=1: Unaceptable a=2: Marginal a=3: Acceptable a=4: Exceeds For all b
Fuzzy membership function boundaries (cross over points) for each of b SoS attributes:	$Bound_{ab}$ a ϵ {1,…h+1}, b ϵ {1,…g} a=1 is lower bound of universe of discourse, a ϵ {2,…h+1} is upper bound of $MF_{(a-1)b}$ because Matlab can't handle matrix subscripts of zero	23	See Table 4.3

Table 4.4. Mathematical definition of variables for ISR domain example (cont.)

Name or description of variable	Expression or Variable Name	Eq. no.	Value for ISR Model
• Overall SoS performance in an Attribute	$(\sum_k^n \sum_i^m A_{ii}\, c_{ki}) * \mathcal{F}$ (A_{ii}, $A_{ij, j\neq i}$,)	24	Flexibility: $\sum_{i=1}^{m}(c_{ij} \times a_{ii}') \geq x$, $x = 0, 1 \ldots m$, where x is the number of systems providing each capability Robustness: (orig perf. – min (perf. stepping through with each different participating system removed))
• Total cost of developing and using an SoS	$TC = \sum_j^n \sum_i^m A_{ij} \mathrm{FIF}_i^{Ss} + \sum_k^n \sum_i^m A_{ii} \mathrm{FOP}_i^{Ss}$	25	Cost = *operations cost* (eq 16) + *development cost (eq. 14)*
Parameters for controlling the netcentric performance factor • Increment per interface • Penalty inc. for unachievable • Penalty dec. for achievable i/f	 Epsilon ϵ Penup Pendn	26	 .02 1 1
Parameters for controlling the GA: • Mutation Rate • Number in Population Number of Generations	 Delta p g	27	 .02 100 50

The binary matrix of capabilities contributed by systems is shown in Figure 4.2. It is equivalent to the x's in the cells on the right side of Figure 4.1. The ISR model with 22 systems is implemented further in the Agent Based Model (ABM) portion of the

FILA-SoS wave development model (Acheson, et al. 2012). Results of the GA operating on each of the domain examples of an SoS introduced in section 4.1 are discussed further in section 4.2.

Capability	CapName	Cap-Sys1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	EO/IR	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	SAR	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
3	Exploit	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
4	C2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
5	Comm	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 4.2. Binary matrix of capabilities vs. systems for ISR example

4.1.2 Operations Other Than War (OOTW) Counterinsurgency ISR

Example. This is a mission with some similarities to the Gulf War ISR mission in Iraq, discussed in section 4.1.1 – not demanding immediate close combat, but more heavily oriented toward surveillance to keep the peace of the assigned area with the possibility of requiring force, but more likely being able to prevent trouble with a show of force. Because the mission and military service is different, the SoS consists of a different set of systems than the Gulf War scenario, with 25 systems and 10 capabilities, with input data as shown in Figure 4.3. Here one can see that some systems have many capabilities, but all still require communications of some sort. The OOTW membership function crossover points are shown in Table 4.5. The OOTW SoS description and characteristics data is shown in

Table 4.6 and Table 4.7. This example was used with one of the operational modeling components of the FILA-SoS project that included scheduling operations and maintenance activities to ensure that the SoS could achieve its mission tasks in a reasonable way.

Name	COIN ISR SOS		For watching an occupied area with counter insurge				A									
NumSys	25		com1	19												
NumCap	10						1	2	3	4	5	6	7	8	9	10
SysNo	**Type**	**Capability**	**I/FDevCos**	**OpsCost/r**	**Perf**	**DevTime**	*EO*	*IR*	*Radar*	*CC Surveil*	*Exploitatic*	*Fusion*	*Coordinat*	*Fires*	*LOS*	*BLOS*
1	Shadow	1	0.01	0.06	85	1	1	0	0	0	0	0	0	0	1	0
2	Shadow	1	0.01	0.06	85	1	1	0	0	0	0	0	0	0	1	0
3	Shadow	2	0.01	0.06	85	1	0	1	0	0	0	0	0	0	1	0
4	Shadow	2	0.01	0.06	85	1	0	1	0	0	0	0	0	0	1	0
5	Shadow	2	0.01	0.06	85	1	0	1	0	0	0	0	0	0	1	0
6	Gray Eagle	3	0.1	0.3	150	1	1	1	1	0	0	0	0	0	1	1
7	Gray Eagle	3	0.1	0.3	150	1	1	1	1	0	0	0	0	0	1	1
8	Apache	8	0	0.2	200	1	1	1	1	0	0	0	0	1	1	0
9	Apache	8	0	0.2	200	1	1	1	1	0	0	0	0	1	1	0
10	CC Surveil	4	0.03	0.09	30	1	0	0	0	1	1	1	1	0	1	1
11	CC Surveil	4	0.03	0.09	30	1	0	0	0	1	1	1	1	0	1	1
12	Exploitati	5	0	0.1	100	1	0	0	0	0	1	1	0	0	0	1
13	Exploitati	6	0	0.1	100	1	0	0	0	0	1	1	0	0	0	1
14	artilllery	8	0.01	0.2	50	1	0	0	0	0	0	0	0	1	1	0
15	UAV Ctrl	7	0.005	0.25	40	1	0	0	0	0	0	0	1	0	1	1
16	UAV Ctrl	7	0.005	0.25	40	1	0	0	0	0	0	0	1	0	1	1
17	Voice/Cha	7	0	0	30	1	0	0	0	0	0	0	1	0	1	1
18	Voice/Cha	7	0	0	30	1	0	0	0	0	0	0	1	0	1	1
19	LOS	9	0.03	0.01	35	1	0	0	0	0	0	0	0	0	1	0
20	LOS	9	0.03	0.01	35	1	0	0	0	0	0	0	0	0	1	0
21	LOS	9	0.03	0.01	35	1	0	0	0	0	0	0	0	0	1	0

Figure 4.3. OOTW IS2 systems and capabilities

Table 4.5. MF edge crossover points for OOTW model

Lower Bound Attributes	**1 Unacceptable**	**1.5 Marginal**	**2.5 Acceptable**	**3.5 Exceeds**	**4 (upper)**
Performance	0.24	0.49	0.63	0.8	1
Affordability	-10	-8	-6	-4	-2
Flexibility	0	1	2	3	4
Robustness	-0.2	-0.15	-0.11	-0.07	-0.03

Table 4.6. OOTW IS2 SoS domain example characteristics

Overarching Purpose of SoS	Peace Keeping ISR in Operations Other Than War
Unique value of SoS	Efficient way to perform the tasks required for Peace keeping
SoS Measures of Effectiveness	Area of territory closely monitored per day Ability to detect and monitor trouble areas early Ability to accurately direct fire or air support to trouble spots

Table 4.6. OOTW IS2 SoS domain example characteristics (cont.)

Issues that might limit effectiveness of the SoS	Bad weather Large number of areas to monitor Deception by enemy forces Ability of guerillas to operate in the civilian community
SoS features that might greatly increase effectiveness	Ability to monitor many areas frequently during both day & night Improved overwatch and backup for patrols/convoys Immediate close air support from armed ISR platforms Communications relay for LOS equipped patrols
Desired Effectiveness	Multiples of full coverage of area of responsibility (AOR) per day Ability to prevent ambush/emplacement of IED in AOR
Stakeholders	Patrolling troops, Local commander(s), System operators, Civil authorities, Higher echelons of command, Local population
ROM Budget: Development	About $40 Million
ROM Budget: Operations	About $40 Million
Attributes of the SoS, and range limits for fuzzy evaluation	Performance – multiples of full coverage of AOR/day Affordability – a few dozens of millions of dollars Robustness – less than 15% loss of capability for loss/absence of one component system Flexibility – prefer no single sources for component capabilities
Capabilities of contributing systems	EO/IR CC Surveillance Fusion Coordination LOS Communications Communications Radar Exploitation Command & Control Fires BLOS

Table 4.7. Mathematical definition of variables for OOTW domain example

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for OOTW Model**
Name of SoS:	sos	1	IS2
Number of potential systems:	m	2	25

Table 4.7. Mathematical definition of variables for OOTW domain example (cont.)

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for OOTW Model**
Number of types of systems:	t	3	10
Names of system types:	$sys\_typ_i : i \in \{1,\ldots t\}$	4	sys_typ1 = Shadow sys_typ2 = Gray Eagle sys_typ3 = Apache sys_typ4 = C&C Surveillance sys_typ5 = Exploitation sys_typ6 = Artillery sys_typ7 = UAV Control sys_typ8 = Voice/Chat sys_typ9 = LOS sys_typ10 = BLOS
Number of component capabilities:	n	5	10
Names of component capabilities:	$sys\_cap_i : i \in \{1,\ldots n\}$	6	sys_cap1 = EO/IR sys_cap2 = SAR sys_cap3 = Exploitation sys_cap4 = Cmd & Control sys_cap5 = Communication
Binary meta-architecture upper triangular matrix:	$A_{ij} : i \in \{1,\ldots m\}, \; j \in \{i,\ldots m\}$	7	Selection of systems and interfaces between them
Individual systems of the SoS	$A_{ij} : i \in \{1,\ldots m\}, \; j = i$, also sometimes written as A_{ii} , or simply A_i	8	Numbered systems up to m=25
Achievable interface	$A_{ij} : i \in \{1,\ldots m\}, \; j > i$, and $A_{jk} = 1$, $A_{ik} = 1$, $A_{ii} = 1$, $A_{jj} = 1$, $A_{kk} = 1$, where A_{kk} is any communications system	9	Depends on both system interfaces with joint communications systems, and systems' presence in the architecture
SoS main capability:	C	10	Detection of insurgency activity
SoS performance in its large capability:	P_{SoS}	11	Expressed as fraction of AOR covered by ISR each half day
Component capabilities of systems:	$c_{ij} : \; i \in \{1,\ldots n\}, \; j \in \{1,\ldots m\}$ (binary)	12	Shown in Figure 4.3. OOTW IS2 systems and capabilities

Table 4.7. Mathematical definition of variables for OOTW domain example (cont.)

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for OOTW Model**
Performance of a particular system in its key capability:	P_i^{Ss} : i ϵ {1,…m}	13	Depends on the system; simplified down to a single gestalt number for this example; Shown in Figure 4.3. OOTW IS2 systems and capabilities
Estimated funding to add an interface to an individual system:	FIF_i^{Ss} : i ϵ {1,…m}	14	Shown in Figure 4.3. OOTW IS2 systems and capabilities
Deadline for developing new interface(s) on a system:	D_i^{Ss} : i ϵ {1,…m}	15	Shown in
Estimated funding for operation of all the participating systems during an SoS operation:	FOP_i^{Ss} : i ϵ {1,…m}	16	Calculated for each chromosome
Function describing the advantage of close collaboration within an SoS as a function of participating systems and interfaces:	F (A_{ii}, A_{ij} , j≠i,) : i ϵ {1,…m}, j ϵ {i,…m}	17	$\sum P_{Systems} * (1+delta)^{(\sum Feas.\ Interfaces\ - \sum Infeas.\ Interfaces)}$
Function for combining system capabilities into SoS capability C:	$C = \sum_k^n \sum_i^m A_{ii}\, c_{ki}$	18	See the Matlab code in Appendix B $\frac{(\sum_i^m a_{ii} P_i \sum_k^n c_{ik})}{\sum_i^m P_i \sum_k^n c_{ik}} (1-\epsilon)^{Penalty}$ Penalty = $(\sum Unachiev.i/f - \sum Achiev.i/f)$
Number of individual attributes the stakeholders want to evaluate the SoS over:	g	19	4
Attribute names to evaluate SoS architectures against (e.g., cost, performance, flexibility):	Att_k : k ϵ {1,…g}	20	Att_1 = Performance Att_2 = Affordability Att_3 = Flexibility Att_4 = Robustness

Table 4.7. Mathematical definition of variables for OOTW domain example (cont.)

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for OOTW Model**
Number of gradations of each Attribute that become Fuzzy Membership Functions (MF):	h_k : k ϵ {1,…g}	21	h_k = 4 for all k
Fuzzy membership function names within each attribute (granulation = a, attribute = b):	MF_{ab} a ϵ {1,…hk}, b ϵ {1,…g}	22	a=1: Unaceptable a=2: Marginal a=3: Acceptable a=4: Exceeds For all b
Fuzzy membership function boundaries (cross over points) for each of b SoS attributes:	$Bound_{ab}$ a ϵ {1,…h+1}, b ϵ {1,…g} a=1 is lower bound of universe of discourse, a ϵ {2,…h+1} is upper bound of $MF_{(a-1)b}$ because Matlab can't handle matrix subscripts of zero	23	See Table 4.6
Overall SoS performance in an Attribute	$(\sum_k^n \sum_i^m A_{ii}\, c_{ki})$ * F (Aii, Aij, j≠i,)	24	Flexibility: $\sum_{i=1}^{m}(c_{ij} \times a_{ii}') \geq x$, x = 0, 1…m, where x is the number of systems providing each capability Robustness: (orig perf. – min (perf. stepping through with each different participating system removed))
Total cost of developing and using an SoS	$TC = \sum_j^n \sum_i^m A_{ij}\mathrm{FIF}_i^{Ss} + \sum_k^n \sum_i^m A_{ii}\mathrm{FOP}_i^{Ss}$	25	Cost = $operations\ cost$ + development cost Ops cost = $\sum_{i=1}^{m} a_i * FOP_i^{Ss}$ Dev cost = $\sum_{i=1}^{m}(\sum_{j=1}^{i-1} a_{ij} * FIF_j^{Ss} + \sum_{j=i+1}^{m} a_{ji} * FIF_j^{Ss})$

Table 4.7. Mathematical definition of variables for OOTW domain example (cont.)

Name or description of variable	Expression or Variable Name	Eq. no.	Value for OOTW Model
Parameters for controlling the netcentric performance factor • Increment per interface • Penalty inc. for unachievable • Penalty dec. for achievable i/f	Epsilon ϵ Penup Pendn	26	.0035 0.5 0.2
Parameters for controlling the GA: • Mutation Rate • Number in Population Number of Generations	Delta P G	27	0.03 40 40

4.1.3 Search and Rescue (SAR) Domain Example. The method as applied in section 4.1 was applied to a non-military ISR domain to insure the fuzzy evaluation and GA would continue to work as hoped. A Coast Guard Search and Rescue (SAR) problem serving the Alaskan coast region was selected. When there is a vessel in distress, the law of the sea requires other mariners to go to its aid, which means that a large number of disparate systems join in an ad hoc SoS. The Coast Guard has numerous systems with differing capabilities such as cutters, aircraft, helicopters, communication systems, and control centers available from several stations in the area. In addition, fishing vessels, civilian craft, and commercial vessels join in this ad hoc SoS to provide assistance when a disaster strikes. To develop improved services in the face of budget cutbacks and changing technologies, it is assumed that adding some communication systems to fishing

boats with their now ubiquitous UAVs to provide better search capability for less total funding. Background information was gathered from numerous Coast Guard documents and news stories about maritime rescues; several SMEs were consulted (Deputy Minister of National Defence and Commissioner, Canadian Coast Guard 1998). A sample SAR SoS with 29 systems of 9 types, with 10 total capabilities, with as many as 9 capabilities per system was constructed as shown in Table 4.9 and Figure 4.6 below. The concept graphic or OV-1 is shown in Figure 4.4.

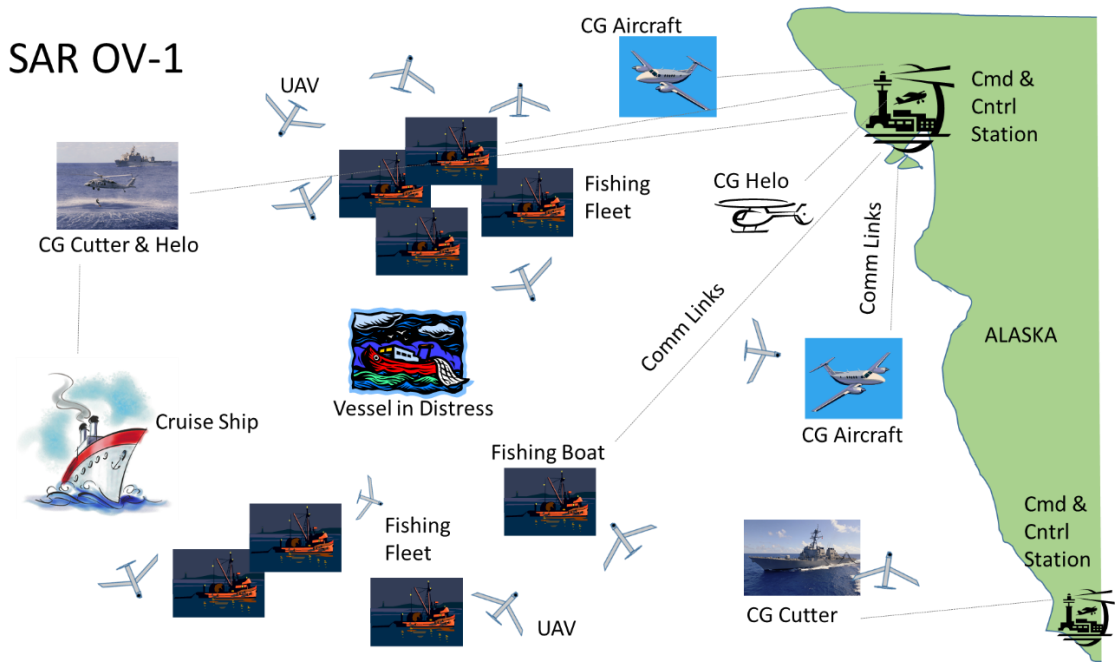


Figure 4.4. Operational View 1 for Search and Rescue scenario

The Search and Rescue (SAR) mission aims to minimize loss of life, injury, and property damage or loss at sea by finding and providing aid to those in distress. The SAR mission framework is inclusive of many activities from conducting search planning and

coordinating SAR response, actual searching for, locating, and rescuing mariners and others in distress, providing necessary medical advice, assistance, or evacuation, and provide, when necessary, persons in distress safe transport to shore. Various components, such as Coast Guard cutters and helicopters, commercial and private sea vessels, Unmanned Vehicles (UVs), and private pilots and aircraft have some reconnaissance capability that may be brought together in a mixed dedicated and ad hoc SoS construct to assist in this ever evolving mission; (Contag, et al. 2013) (Johnston, et al. 2013).

> "As defined in the National Search and Rescue Plan, ref (a), and Supplement, ref (b), participating search and rescue organizations may obtain permissible assets within the required SAR regions at any notice. These regions include all waters subject to U.S. jurisdiction and international waters in the Atlantic, Pacific, and Arctic Oceans and the Gulf of Mexico. Additional regions include identified Department of Defense (DoD) Area of Responsibilities (AORs). Partnerships exist among maritime industry in the Automated Mutual-Assistance Vessel Rescue (AMVER) system, and coordination among Federal, state, local, and tribal authorities to coordinate SAR operations is extensive. This section describes an example operational context for SAR missions, for which optimal SoS configurations can be determined given specific mission parameters and tradeoffs among SoS attributes such as performance, flexibility, robustness, and affordability."

Use of the Bering Sea and the Arctic by commercial fisheries, oil exploration, ecology and climate science is increasing. With the rise of the number of people and vessels in the area, the likelihood increases of a large SAR scenario occurring. Possible missions related to this setting may include those in Table 4.8. The corresponding domain information overview is shown in Table 4.9, MF crossover points in Table 4.10, mathematical model definitions in Table 4.11, and computer data input in Figure 4.6 below.

Table 4.8. Possible SAR scenarios

	Possible SAR Scenarios
1	A large sinking ship, cruise liner, or commercial freighter. Rescue of passengers, and/or a potential exposure of hazardous material (oil).
2	A ship stuck in the ice in the arctic ocean.
3	A private or commercial plane crash with survivors.
4	An oil rig disaster (fire, explosion, medical emergency, etc.).

The basic conceptual radius of operation for the purposes of this application will include the Bearing Sea and the Gulf of Alaska as represented in Figure 4.5 below. This is the actual area of responsibility of the US Coast Guard District 17. Evolving extended loiter radii for airborne ISR mission profiles may extend the conceptual SAR mission profile to include the North Pacific Ocean, Chukchi Sea, Beaufort Sea, and Arctic Ocean. Scientific expeditions and mineral exploitation efforts are growing in this larger area as well, so this is a useful exercise.

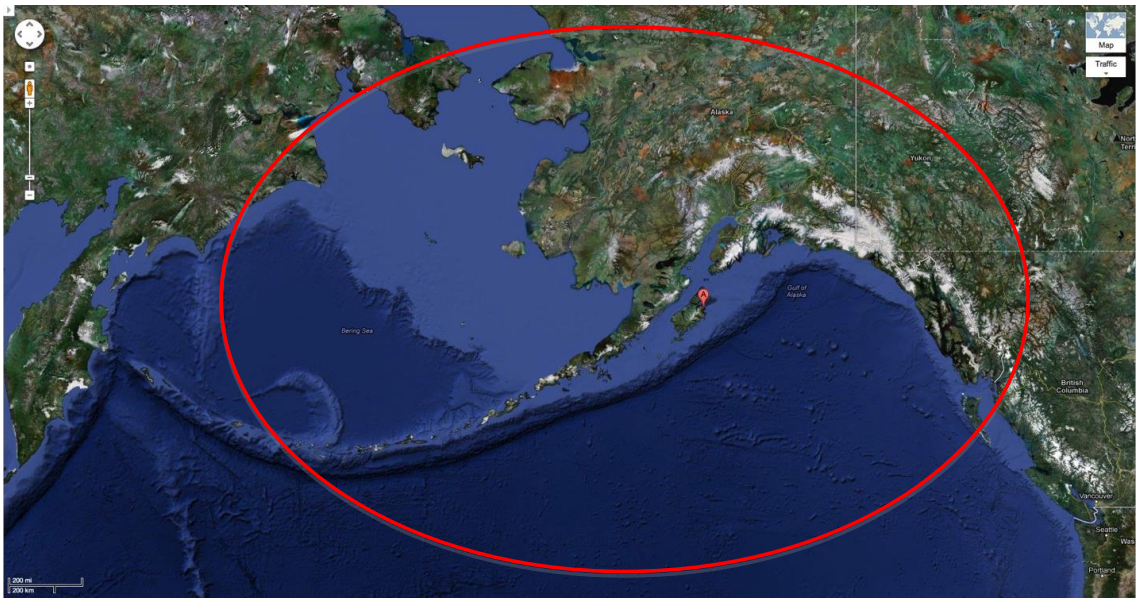


Figure 4.5. Conceptual SAR Operating Radius (Google Maps, 2013)

Table 4.9. Characteristics of a SAR SoS

Overarching Purpose of SoS	Maritime Search & Rescue (SAR) of Bering Sea; small airliner crash at sea Stranded Cruise Ship in 'Other Territorial' Waters Find two people in a small boat
Unique value of SoS	Greatly enhanced SAR Capability
SoS Measures of Effectiveness	Time to search 100,000 Sq Mi Probability of detection of survivors within 2 hours or within 12 hours, depending on the scenario
Issues that might limit effectiveness of the SoS	Weather Availability of participant systems Language barriers Number of Survivors Sovereignty questions
SoS features that might greatly increase effectiveness	Speed of discovery Improved coordination of resources Ability to prioritize resources(?) at time of event, or during development
Desired Effectiveness	Find someone very fast and/or help lots of people relatively fast

Table 4.9. Characteristics of a SAR SoS (cont.)

Stakeholders	Federal, State, Local, Tribal governments NGOs, Foreign Nation, Crews, Mariners, travel/shipping/fishing/oil/research/insurance corporations, Survivors, Military, Coast Guard, Public
ROM Budget: Development	Around $15M
ROM Budget: Operations	Around $10M
Attributes of the SoS, and range limits for fuzzy evaluation	Performance – time to find and pick someone up before death by exposure or injury Affordability – budgetary pressures, small civilian investment Robustness – still works with only partial complement of systems Flexibility - many choices of partners
Capabilities of contributing systems	EO/IR Night Vision Maritime Radar Emergency Locator Beacon System Tracking RF direction finder Deliver Paramedic/medical aid Remove survivor(s) to Emergency Medical Care Provide major medical capability Speed – Fast (around 300 kt)/Slow (around 15 kt) Time on Station Command and Control/Coordination Communications

Costs for developing the interfaces are assigned to each system, as well as a cost for operating the system for a month in the case of the ISR SoS, or for 3 days in the case of the SAR SoS. The deadline for development of an interface was assigned one of three values:

- 0 – ready now,
- 1 – will be ready by the end of this epoch, or
- 2 – won't be ready this epoch, but the next.

Name	SAR							A								
NumSys	29															
NumCap	10						1	2	3	4	5	6	7	8	9	10
SysNo	**Type**	**Capability**	**I/FDevCos**	**OpsCost/I**	**Perf**	**DevTime**	*IR – range*	*Night Visi*	*Visual – ra*	*Maritime*	*RF Directi*	*Deliver Me*	*Remove s*	*Speed 300*	*Speed 15 r*	*Communi*
1	Cutter	7	0.03	2	12	1		x	x	x	x	x	x		x	x
2	Cutter	7	0.03	2	12	1		x	x	x	x	x	x		x	x
3	Helicopte	6	0.1	2	20	1	x	x	x	x	x	x	x	x		x
4	Helicopte	6	0.1	2	20	1	x	x	x	x	x	x	x	x		x
5	Aircraft	8	0.1	5	10	1			x		x			x		x
6	Aircraft	8	0.1	5	10	1			x		x			x		x
7	UAV	1	0.1	0.1	7	1	x		x	x			x		x	x
8	UAV	1	0.1	0.1	7	1	x		x	x			x		x	x
9	UAV	1	0.1	0.1	7	1	x		x	x			x		x	x
10	UAV	1	0.1	0.1	7	1	x		x	x			x		x	x
11	UAV	1	0.1	0.1	7	1	x		x	x			x		x	x
12	UAV	1	0.1	0.1	7	1	x		x	x			x		x	x
13	UAV	1	0.1	0.1	7	1	x		x	x			x		x	x
14	UAV	3	0.1	0.1	7	1	x		x	x			x		x	x
15	UAV	3	0.1	0.1	7	1	x		x	x			x		x	x
16	UAV	3	0.1	0.1	7	1	x		x	x			x		x	x
17	UAV	3	0.1	0.1	7	1	x		x	x			x		x	x
18	Fish Vesse	3	0.03	0.5	4	1			x	x		x	x		x	x
19	Fish Vesse	3	0.03	0.5	4	1			x	x		x	x		x	x
20	Fish Vesse	3	0.03	0.5	4	1			x	x		x	x		x	x
21	Fish Vesse	3	0.03	0.5	4	1			x	x		x	x		x	x
22	Fish Vesse	3	0.03	0.5	4	1			x	x		x	x		x	x
23	Civ Ship	7	0.05	2	8	1			x	x		x	x		x	x
24	Coord Ctr	5	0.05	0.5	5	1					x	x			x	x
25	Coord Ctr	5	0.05	0.5	5	1					x	x			x	x
26	Communi	10	0.02	0.03	1	0										x
27	Communi	10	0.02	0.03	1	0										x
28	Communi	10	0.02	0.03	1	0										x
29	Communi	10	0.02	0.03	1	0										x

Figure 4.6. The fuzzy assessor model inputs for the SAR SoS

Table 4.10. MF edge crossover points for SAR

Lower Bound Attributes	**1 Unacceptable**	**1.5 Marginal**	**2.5 Acceptable**	**3.5 Exceeds**	**4 (upper)**
Performance	0	0.12	0.24	0.36	0.45
Affordability	-50	-40	-33	-22	-10
Flexibility	0	1	2	3	4
Robustness	-0.25	-0.18	-0.12	-0.06	-0.01

Table 4.11. Mathematical definitions for SAR model

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for SAR Model**
Name of SoS:	sos	1	SAR
Number of potential systems:	m	2	29
Number of types of systems:	t	3	8

Table 4.11. Mathematical definitions for SAR model (cont.)

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for SAR Model**
Names of system types:	$sys\_typ_i : i \in \{1,\ldots t\}$	4	sys_typ1 = Cutter sys_typ2 = Helicopter sys_typ3 = Aircraft sys_typ4 = UAV sys_typ5 = Fish Vessel sys_typ6 = Civ Ship sys_typ7 = Coord Ctr sys_typ8 = Communications
Number of component capabilities:	n	5	10
Names of component capabilities:	$sys\_cap_i : i \in \{1,\ldots n\}$	6	sys_cap1 = IR sys_cap2 = Night Vision sys_cap3 = Visual sys_cap4 = Maritime Radar sys_cap5 = RF Dir Find sys_cap6 = Deliver Med Care sys_cap7 = Remove Survivor sys_cap8 =Speed 300 kt sys_cap9 =Speed 15 kt sys_cap10 = Communications
Binary meta-architecture upper triangular matrix:	$A_{ij} : i \in \{1,\ldots m\}, \; j \in \{i,\ldots m\}$	7	Selection of systems and interfaces between them
Individual systems of the SoS	$A_{ij} : i \in \{1,\ldots m\}, \; j = i$, also sometimes written as A_{ii} , or simply A_i	8	Numbered systems up to m=29
Achievable interface	$A_{ij} : i \in \{1,\ldots m\}, \; j > i$, and $A_{jk} = 1$, $A_{ik} = 1$, $A_{ii} = 1$, $A_{jj} = 1$, $A_{kk} = 1$, where A_{kk} is any communications system	9	Depends on both system interfaces with joint communications systems, and systems' presence in the architecture
SoS main capability:	C	10	Find and rescue survivors
SoS performance in its large capability:	P_{SoS}	11	Torrent problem toy problem Expressed as probability of finding a survivor within 2-12 hours in frigid temps
Component capabilities of systems:	$c_{ij} : i \in \{1,\ldots n\}$, $j \in \{1,\ldots m\}$ (binary)	12	Shown in Figure 4.6

Table 4.11. Mathematical definitions for SAR model (cont.)

Name or description of variable	Expression or Variable Name	Eq. no.	Value for SAR Model
Performance of a particular system in its key capability:	P_i^{Ss} : i ϵ {1,…m}	13	Depends on the system; simplified down to a single gestalt number for this example; shown in Figure 4.6
Estimated funding to add an interface to an individual system:	FIF_i^{Ss} : i ϵ {1,…m}	14	Shown in Figure 4.6
Deadline for developing new interface(s) on a system:	D_i^{Ss} : i ϵ {1,…m}	15	Shown in Figure 4.6
Estimated funding for operation of all the participating systems during an SoS operation:	FOP_i^{Ss} : i ϵ {1,…m}	16	Calculated for each chromosome
Function describing the advantage of close collaboration within an SoS as a function of participating systems and interfaces:	F (A_{ii}, A_{ij} , j≠i,) : i ϵ {1,…m}, j ϵ {i,…m}	17	$\sum P_{Systems} * (1+delta)^{(\sum Achiev.\ Interfaces - \sum Unachiev.\ Interfaces)}$
Function for combining system capabilities into SoS capability C:	$C = \sum_k^n \sum_i^m A_{ii}\, c_{ki}$	18	$\frac{(\sum_i^m a_{ii} P_i \sum_k^n c_{ik})}{\sum_i^m P_i \sum_k^n c_{ik}} * (1-\epsilon)^{Penalty}$ Penalty = $(\sum Unachiev.i/f - \sum Achiev.i/f)$
Number of individual attributes the stakeholders want to evaluate the SoS over:	g	19	4
Attribute names to evaluate SoS architectures against (e.g., cost, performance, flexibility):	Att_k : k ϵ {1,…g}	20	Att_1 = Performance Att_2 = Affordability Att_3 = Flexibility Att_4 = Robustness

Table 4.11. Mathematical definitions for SAR model (cont.)

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for SAR Model**
Number of gradations of each Attribute that become Fuzzy Membership Functions (MF):	h_k : k ϵ {1,…g}	21	h_k = 4 for all k
Fuzzy membership function names within each attribute (granulation = a, attribute = b):	MF_{ab} a ϵ {1,…hk}, b ϵ {1,…g}	22	a=1: Unaceptable a=2: Marginal a=3: Acceptable a=4: Exceeds For all b
Fuzzy membership function boundaries (cross over points) for each of b SoS attributes:	$Bound_{ab}$ a ϵ {1,…h+1}, b ϵ {1,…g} a=1 is lower bound of universe of discourse, a ϵ {2,…h+1} is upper bound of $MF_{(a-1)b}$ because Matlab can't handle matrix subscripts of zero	23	See Table 4.10. MF edge crossover points for SAR
Overall SoS performance in an Attribute	$(\sum_k^n \sum_i^m A_{ii}\, c_{ki})$ * F (Aii, Aij, j≠i,)	24	$\frac{(\sum_i^m a_{ii} P_i \sum_k^n c_{ik} + P_i^{Ss} c_{ik})}{\sum_i^m P_i \sum_k^n c_{ik}} * (1-\epsilon)^{Penalty}$ Penalty = $(\sum Unachiev.i/f - \sum Achiev.i/f)$
• Total cost of developing and using an SoS	$TC = \sum_j^n \sum_i^m A_{ij} \mathrm{FIF}_i^{Ss} + \sum_k^n \sum_i^m A_{ii} \mathrm{FOP}_i^{Ss}$	25	Ops cost = $\sum_{i=1}^m a_i * FOP_i^{Ss}$ Dev cost = $\sum_{i=1}^m (\sum_{j=1}^{i-1} a_{ij} * FIF_j^{Ss} + \sum_{j=i+1}^m a_{ji} * FIF_j^{Ss})$ Cost = operations cost + development cost

Table 4.11. Mathematical definitions for SAR model (cont.)

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for SAR Model**
Parameters for controlling the netcentric performance factor			
• Increment per interface	Epsilon ϵ	26	0.008 & 0.005 & 0.01
• Penalty inc. for unachievable	Penup		0.4 0.3 0.3
• Penalty dec. for achievable i/f	Pendn		0.6 0.2 0.8
Parameters for controlling the GA:			
• Mutation Rate	Delta	27	0.02 & 0.005 & 0.01
• Number in Population	p		80 80 300
Number of Generations	g		50 50 50

A system may spend funds on an interface that will not be ready until the next epoch, but they will get no performance increment from that interface until it is complete. An overall 'relative' performance value was assigned to each system based on its key capability. The costs for development were rough figures similar to what may be seen in official and informal budgetary estimates for interfacing with communications systems and integrating the mission systems to be able to interoperate. The costs to operate aircraft or other systems were determined similarly, in units of thousands of dollars per flight hour. The units are chosen to result in numbers usually between 0.1 to 100 because it makes comparisons more intuitive and easier to keep straight in one's head.

4.1.3.1 A model building basis for SAR. New tools are being developed that could make the integration of the SoS exploration and analysis tools developed here even easier to use. When building the SAR model, autogenerating the domain input data from a more general descriptive model of a system or SoS was examined. It does appear possible, but additional development would be required. The activity diagram in Figure 4.8, built using classes that equate to the types of systems used in SAR, is an example of the way that today's SoS architects are being taught at the Naval Postgraduate School. This is the way analysts and graduate systems engineers are being trained to think and communicate architecture concepts among themselves and to others. This relatively new tool can already autogenerate an execution timeline such as that shown in Figure 4.7 (SPEC Innovations 2015). The point of this is not to recommend a tool, but to note that newer tools are evolving to be able to support the types of representation and analysis that will make architecting future SoS far more effective and efficient. Competitive pressure will move all the tool vendors in this direction.

4.1.3.2 Additional features of recent tool versions. Multiple executions can be set up in Monte Carlo simulations to obtain analysis statistics of a model architecture as well. This type of connection between tools, architectures and analysis might be fruitful to pursue in future work. The activity diagram shown in Figure 4.8 shows swim lanes, serial step sequences, data exchanges between steps, loops, and parallel paths.

4.1.4 MITRE "Toy" Problem. MITRE presents what they call the Toy SoS problem that has been studied fairly extensively within the government (DeLaurentis, et al. 2012) and academically (Guariniello and DeLaurentis 2014). This SoS problem was

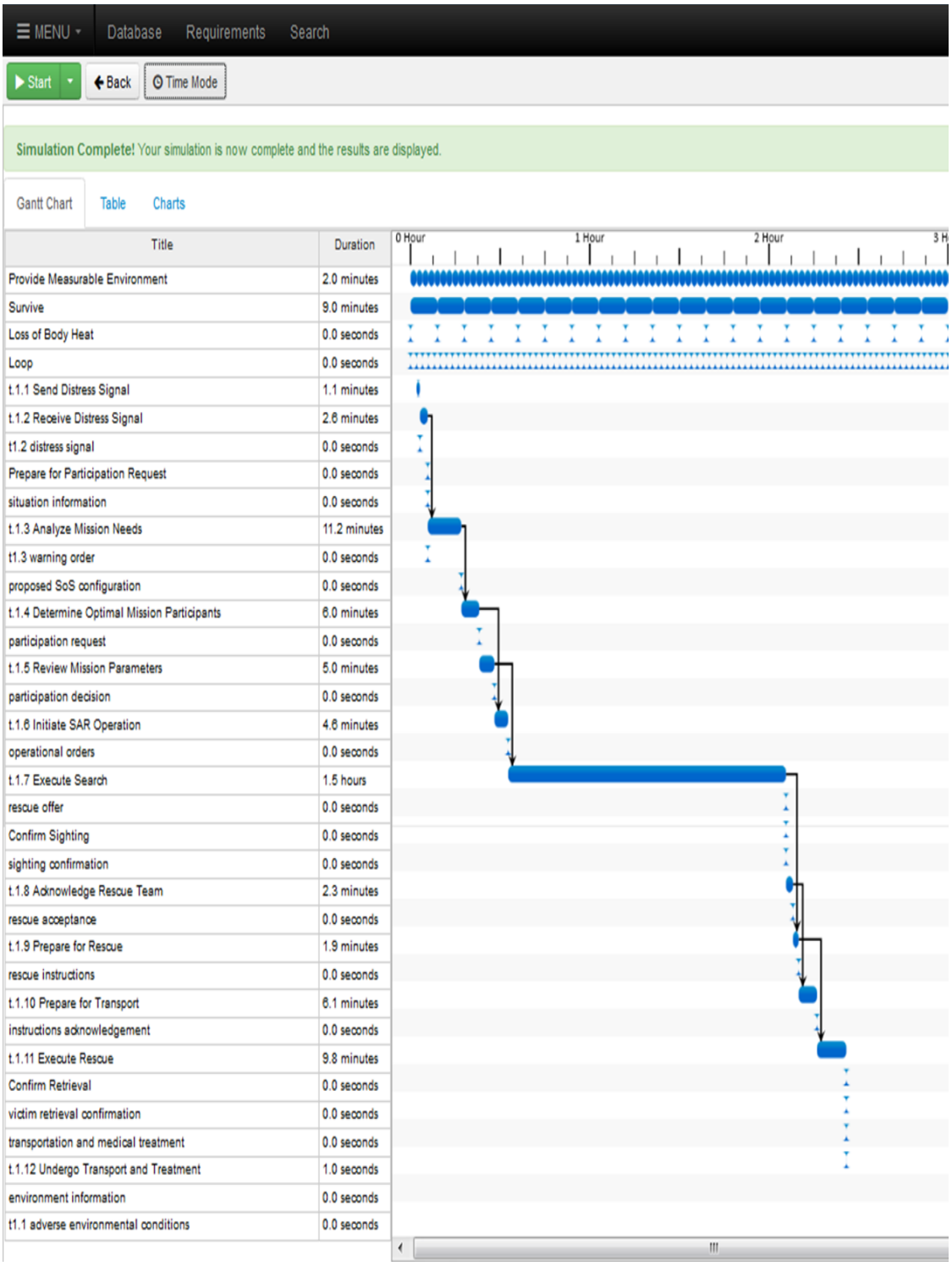


Figure 4.7. Execution timeline example generated directly from the SAR model

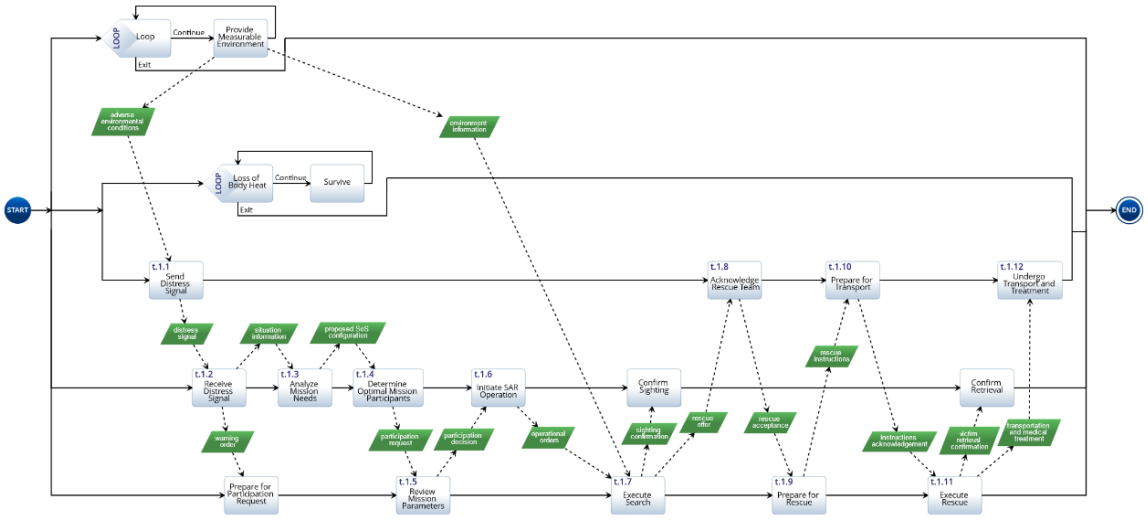


Figure 4.8. Activity diagram matching the CONOPS of the SAR model

recast in the format used in FILA-SoS, but the original Toy problem in Figure 4.9 is too small to work properly in FILA-SoS, because all component systems must be included and they all have only one capability in the original formulation – there is nothing to select. Therefore, there is really no opportunity to trade different numbers of system types or combinations of systems and interfaces among themselves as FILA-SoS does. Additionally, the network connection graph is *directed* in the Toy problem, whereas in FILA-SoS only undirected graphs were used. Finally, the performance attribute in the Toy problem was calculated using the functional dependency network analysis algorithm, so a very different form of input domain data is required (Garvey and Pinto 2009). FDNA assumes the links are always associated with each system, not counted separately as in FILA-SoS, but it adds to the model by including a 'criticality of dependency' (COD) and a 'strength of dependency' (SOD) value for each link.

For purposes of having a few more systems to choose from, the Toy problem was initially reconfigured as shown in Figure 4.10. The corresponding input domain data is shown in Figure 4.11. The additional Missouri modification input COD and SOD data is on pages 316-317 of Appendix E. The MF data shown in Table 4.12 uses the ratio of original COD data to the COD as the key measure when systems are reduced in efficiency by maintenance failure or by attack. The remaining Toy problem data is shown in Table 4.13 and Table 4.14. The affordability MF limits are set in this version so that too many or too few systems will be discarded from the solution by the GA.

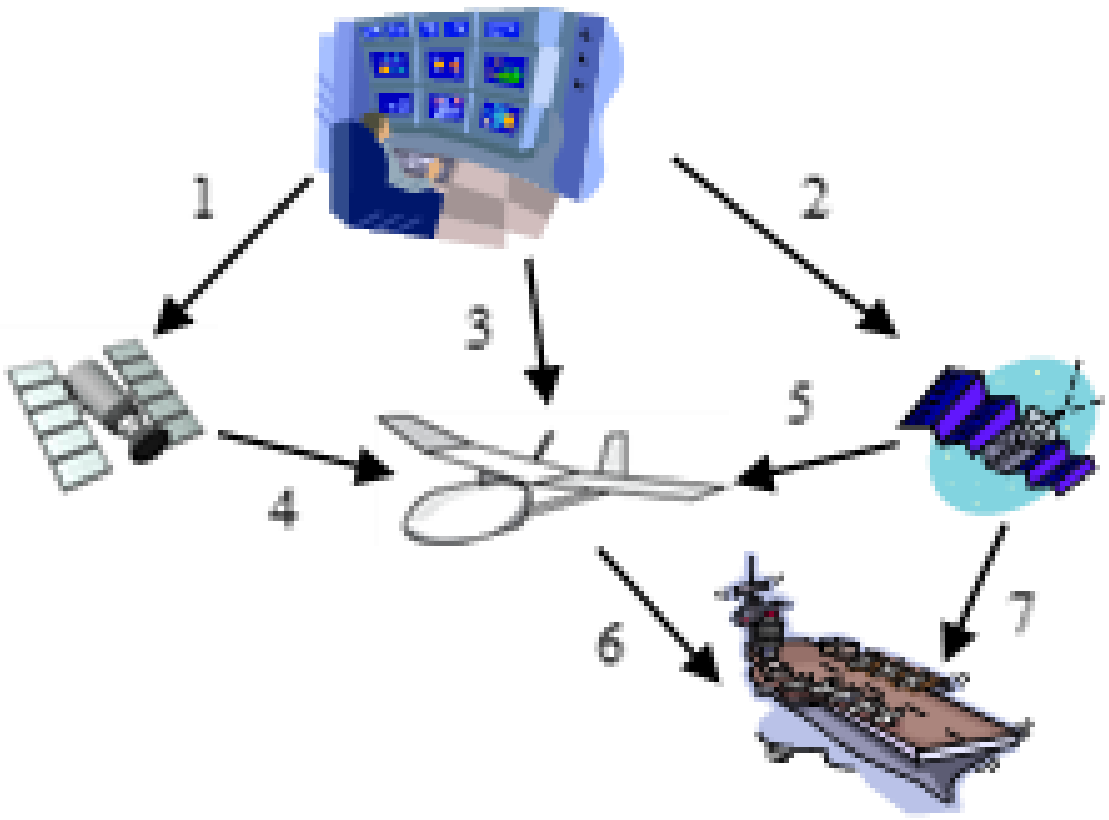


Figure 4.9. MITRE Toy SoS problem as originally proposed

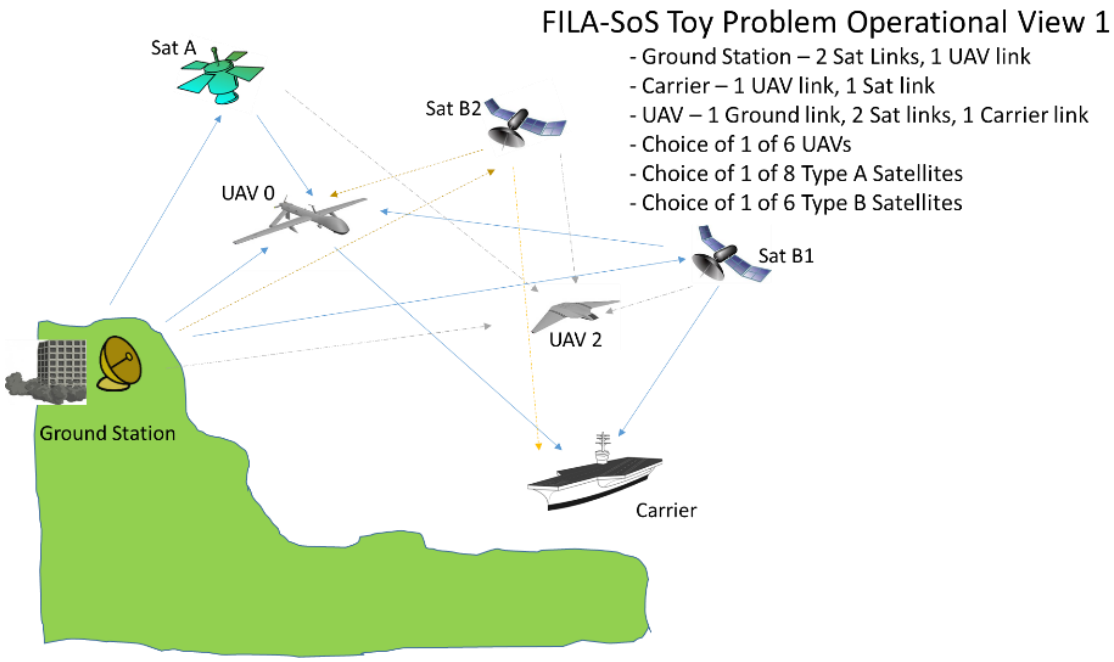


Figure 4.10. Reconfigured Toy problem for Missouri Toy FILA-SoS approach

Name	TOY							A			
NumSys	22		com1	23							
NumCap	5		*sys has capability, costs, perf, deadline*				1	2	3	4	5
SysNo	**Type**	**Capability**	**I/FDevCos**	**OpsCost/I**	**Perf**	**DevTime**	Ground	SatA	UAV	SatB	Carrier
1	Ground	1	0	1	100	0	x				
2	SatA1	2	0	1	100	0		x			
3	SatA2	2	0	1	100	0		x			
4	SatA3	2	0	1	100	0		x			
5	SatA4	2	0	1	100	0		x			
6	SatA5	2	0	1	100	0		x			
7	SatA6	2	0	1	100	0		x			
8	SatA7	2	0	1	100	0		x			
9	SatA8	2	0	1	100	0		x			
10	UAV0	3	0	1	100	0			x		
11	UAV1	3	0	1	100	0			x		
12	UAV2	3	0	1	100	0			x		
13	UAV3	3	0	1	100	0			x		
14	UAV4	3	0	1	100	0			x		
15	UAV5	3	0	1	100	0			x		
16	SatB1	4	0	1	100	0				x	
17	SatB2	4	0	1	100	0				x	
18	SatB3	4	0	1	100	0				x	
19	SatB4	4	0	1	100	0				x	
20	SatB5	4	0	1	100	0				x	
21	SatB6	4	0	1	100	0				x	
22	Carrier	5	0	1	100	0					x

Figure 4.11. Input domain data for FILA-SoS configured Toy problem

Table 4.12. MF edge crossover points for TOY problem

Lower Bound Attributes	1	1.5	2.5	3.5	4
Performance Ratio	0	0.8	0.9	0.98	1
Affordability	-50	-6	-5.5	-5	-4.8
Flexibility	0	0.25	0.5	1	2
Robustness	-0.25	-0.18	-0.12	-0.06	-0.01

Table 4.13. MITRE Toy problem SoS domain datasheet

Overarching Purpose of SoS	Relay commands and ISR data from ground station and UAV to a Carrier Battle Group
Unique value of SoS	Provide redundant paths for data important to the Carrier Battle Group (CBG)
SoS Measures of Effectiveness	Reliability of data links Latency of data
Issues that might limit effectiveness of the SoS	Weather Availability of participant systems Cyber attacks on elements of system Jamming of communications links
SoS features that might greatly increase effectiveness	Similarity of data link formatting Over the horizon communications links Frequency diversity Redundant messages
Desired Effectiveness	99.999% up time for end to end communications Full bandwidth availability
Stakeholders	Carrier Battle Group Users Satellite operators UAV controller UAV owner Information Generators Other potential Users of links Ground Station Operators
ROM Budget: Development	About $10M
ROM Budget: Operations	About $5M

Table 4.13. MITRE Toy problem SoS domain datasheet (cont.)

Attributes of the SoS, and range limits for fuzzy evaluation	Performance – redundancy of communications links (individual links all perform the same in Toy problem) Affordability – budgetary pressures, small investment (basically all acceptable for Toy problem) Robustness – still works with only partial complement of systems Flexibility - many choices of partners
Capabilities of contributing systems	Ground station uplinks Relay capability of satellites Relay capability of UAV Receive capability of Carrier Battle Group

Table 4.14. Mathematical definition of variables for Missouri Toy problem

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for Toy Model**
Name of SoS:	sos	1	TOY
Number of potential systems:	m	2	22
Number of types of systems:	t	3	5
Names of system types:	$sys\_typ_i : i \in \{1,\ldots t\}$	4	sys_typ1 = Ground sys_typ2 = SatAx, x ϵ (1 - 8) sys_typ3 = UAV x, x ϵ (0 - 5) sys_typ4 = SatB x, x ϵ (1 - 6) sys_typ5 = Carrier
Number of component capabilities:	n	5	5
Names of component capabilities:	$sys\_cap_i : i \in \{1,\ldots n\}$	6	sys_cap1 = Ground sys_cap2 = SatA sys_cap3 = UAV sys_cap4 = SatB sys_cap5 = Carrier
Binary meta-architecture upper triangular matrix:	$A_{ij} : i \in \{1,\ldots m\}, j \in \{i,\ldots m\}$	7	Selection of systems and interfaces between them
Individual systems of the SoS	$A_{ij} : i \in \{1,\ldots m\}, j = i$, also sometimes written as A_{ii}, or simply A_i	8	Numbered systems up to m=22

Table 4.14. Mathematical definition of variables for Missouri Toy problem (cont.)

Name or description of variable	Expression or Variable Name	Eq. no.	Value for Toy Model
Achievable interface	A_{ij} : i ϵ {1,…m}, j > i , and A_{jk} = 1, A_{ik} = 1, A_{ii} =1, A_{jj}=1, A_{kk} = 1 , where A_{kk} is any communications system	9	All achievable except the Ground system does not interface with the Carrier, and systems of type SatA do not interface with type Sat B
SoS main capability:	C	10	Performance ratio of Connection Ground to Carrier
SoS performance in its large capability:	P_{SoS}	11	Continuity of connection
Component capabilities of systems:	c_{ij} : i ϵ {1,…n}, j ϵ {1,…m} (binary)	12	Whether each system posseses each capability
Performance of a particular system in its key capability:	P_i^{Ss} : i ϵ {1,…m}	13	COD and SOD for each system are shown on page E-13 of Appendix E for initial solution; the general solution used COD/SOD in Figure 64-65
Estimated funding to add an interface to an individual system:	FIF_i^{Ss} : i ϵ {1,…m}	14	Shown in Figure 4.1; all the same
Deadline for developing new interface(s) on a system:	D_i^{Ss} : i ϵ {1,…m}	15	Shown in Figure 4.1; all the same
Estimated funding for operation of all the participating systems during an SoS operation:	ΣFOP_i^{Ss} : i ϵ {1,…m}	16	Calculated for each chromosome's selected systems
Function describing the advantage of close collaboration within an SoS as a function of participating systems and interfaces:	F (A_{ii}, A_{ij} , j≠i,) : i ϵ {1,…m}, j ϵ {i,…m}	17	FDNA implementation of COD and SOD matrices
Function for combining system capabilities into SoS capability C:	$C = \sum_k^n \sum_i^m A_{ii}\, c_{ki}$	18	See the Matlab code in Appendix B pg 21, file evalsos.m and fdn22Atoy.m for the Toy problem

Table 4.14. Mathematical definition of variables for Missouri Toy problem (cont.)

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for Toy Model**
Number of individual attributes the stakeholders want to evaluate the SoS over:	g	19	1; other attributes are used only to select out undesired chromosomes for initial solution 4 were used in the second, general FDNA implementation
Attribute names to evaluate SoS architectures against (e.g., cost, performance, flexibility):	Att_k : $k \in \{1,\ldots g\}$	20	Att_1 = PerformanceRatio (before and after attacks); same for both implementations Att_2 = Affordability Att_3 = SinglePtFailure Att_4 = StrengthOfDepen
Number of gradations of each Attribute that become Fuzzy Membership Functions (MF):	h_k : $k \in \{1,\ldots g\}$	21	$h_k = 4$ for all k
Fuzzy membership function names within each attribute (granulation = a, attribute = b):	MF_{ab} $a \in \{1,\ldots hk\}$, $b \in \{1,\ldots g\}$	22	a=1: Unaceptable a=2: Mediocre a=3: AboveAvg a=4: VeryGood For all b
Fuzzy membership function boundaries (cross over points) for each of b SoS attributes:	$Bound_{ab}$ $a \in \{1,\ldots h+1\}$, $b \in \{1,\ldots g\}$ a=1 is lower bound of universe of discourse, $a \in \{2,\ldots h+1\}$ is upper bound of $MF_{(a-1)b}$ because Matlab can't handle matrix subscripts of zero	23	See Table 4.12. MF edge crossover points for TOY problem
Overall SoS performance in an Attribute	$(\sum_k^n \sum_i^m A_{ii}\, c_{ki})$ * F (Aii, Aij, j≠i,)	24	See the Matlab code in Appendix B pg 21; it is unique for the generalized Toy problem

Table 4.14. Mathematical definition of variables for Missouri Toy problem (cont.)

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for Toy Model**
Total cost of developing and using an SoS	$TC = \sum_j^n \sum_i^m A_{ij} \mathrm{FIF}_i^{Ss} + \sum_k^n \sum_i^m A_{ii} \mathrm{FOP}_i^{Ss}$	25	See the Matlab code in Appendix B; used only to confirm a feasible chromosome in first solution, when only one of each type is chosen (no costs); General formulation of FDNA evaluator included multiple systems of each type, so cost counted
Parameters for controlling the netcentric performance factor • Increment per interface • Penalty inc for unachievable • Penalty decrement for achievable i/f	Epsilon ϵ Penup Pendn	26	N/A N/A N/A
Parameters for controlling the GA: • Mutation Rate • Number in Population • Number of Generations	Delta p g	27	0.02 40 50

4.1.5 Large Live-Virtual-Constructive (LVC) Model. MITRE and the Army supplied a very large, proprietary training SoS problem for validation of the method on a realistic problem. It was broken down to 111 systems with 74 capabilities after exploring architecture description information from relatively complete DoDAF compliant information on the SoS in several proprietary documents and stakeholder summary

presentations. A sanitized version of the SoS domain data is shown in Table 4.15, with the mathematical definitions for the LVC problem in Table 4.16.

Table 4.15. MITRE Proprietary LVC problem SoS domain datasheet

Overarching Purpose of SoS	Enable, and enhance the value of, Live, Virtual, Constructive training
Unique value of SoS	Allows many existing automated and operator in the loop training simulations, and live participants to train simultaneously
SoS Measures of Effectiveness	Proprietary
Issues that might limit effectiveness of the SoS	Latency Mistranslation of data between different systems Lack of centralized truth data
SoS features that might greatly increase effectiveness	Establishment of central truth data Common interfaces among participating systems Improved sense of reality to training simulations Allowing any mix of live, virtual, and constructive participants
Desired Effectiveness	Proprietary
Stakeholders	Trainers and trainees (both users of the systems), funders, system developers
ROM Budget: Development	Proprietary
ROM Budget: Operations	Proprietary
Attributes of the SoS, and range limits for fuzzy evaluation	Seven Proprietary attributes
Capabilities of contributing systems	Communications Displays Simulations of numerous tasks to be trained

Table 4.16. Mathematical definition of variables for LVC validation problem

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for LVC Model**
Name of SoS:	sos	1	LVC
Number of potential systems:	m	2	111
Number of types of systems:	t	3	18
Names of system types:	$sys\_typ_i : i \in \{1,\ldots t\}$	4	Proprietary
Number of component capabilities:	n	5	74
Names of component capabilities:	$sys\_cap_i : i \in \{1,\ldots n\}$	6	Proprietary
Binary meta-architecture upper triangular matrix:	$A_{ij} : i \in \{1,\ldots m\}, j \in \{i,\ldots m\}$	7	Selection of systems and interfaces between them
Individual systems of the SoS	$A_{ij} : i \in \{1,\ldots m\}, j = i$, also sometimes written as A_{ii}, or simply A_i	8	Numbered systems up to m=111
Achievable interface	$A_{ij} : i \in \{1,\ldots m\}, j > i$, and $A_{jk} = 1, A_{ik} = 1, A_{ii} = 1, A_{jj}=1, A_{kk} = 1$, where A_{kk} is any communications system	9	Proprietary
SoS main capability:	C	10	Training
SoS performance in its large capability:	P_{SoS}	11	Training effectiveness
Component capabilities of systems:	$c_{ij} : i \in \{1,\ldots n\}, j \in \{1,\ldots m\}$ (binary)	12	Whether each system posseses each capability
Performance of a particular system in its key capability:	$P_i^{Ss} : i \in \{1,\ldots m\}$	13	Proprietary
Estimated funding to add an interface to an individual system:	$FIF_i^{Ss} : i \in \{1,\ldots m\}$	14	Proprietary
Deadline for developing new interface(s) on a system:	$D_i^{Ss} : i \in \{1,\ldots m\}$	15	Proprietary

Table 4.16. Mathematical definition of variables for LVC validation problem (cont.)

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for LVC Model**
Estimated funding for operation of all the participating systems during an SoS operation:	ΣFOP_i^{Ss} : i ϵ {1,…m}	16	Calculated for each chromosome's selected systems
Function describing the advantage of close collaboration within an SoS as a function of participating systems and interfaces:	F (A_{ii}, A_{ij} , j≠i,) : i ϵ {1,…m}, j ϵ {i,…m}	17	Not Used
Function for combining system capabilities into SoS capability C:	$C = \sum_k^n \sum_i^m A_{ii}\, c_{ki}$	18	See the Matlab code in Appendix B for LVC problem
Number of individual attributes the stakeholders want to evaluate the SoS over:	g	19	7
Attribute names to evaluate SoS architectures against (e.g., cost, performance, flexibility):	Att_k : k ϵ {1,…g}	20	Proprietary
Number of gradations of each Attribute that become Fuzzy Membership Functions (MF):	h_k : k ϵ {1,…g}	21	h_k = 5 for all k
Fuzzy membership function names within each attribute (granulation = a, attribute = b):	MF_{ab} a ϵ {1,…hk}, b ϵ {1,…g}	22	Proprietary

Table 4.16. Mathematical definition of variables for LVC validation problem (cont.)

Name or description of variable	**Expression or Variable Name**	**Eq. no.**	**Value for LVC Model**
Fuzzy membership function boundaries (cross over points) for each of b SoS attributes:	$Bound_{ab}$ a ϵ {1,…h+1}, b ϵ {1,…g} a=1 is lower bound of universe of discourse, a ϵ {2,…h+1} is upper bound of $MF_{(a-1)b}$ because Matlab can't handle matrix subscripts of zero	23	Proprietary
Overall SoS performance in an Attribute	$(\sum_k^n \sum_i^m A_{ii}\, c_{ki})$ * F (A_{ii}, A_{ij}, j≠i,)	24	See the Matlab code in Appendix B; it is unique for the LVC problem; e.g., AU = $\sum_{i=60}^{90}(\sum_{j=2}^{53} a_{ii}\, c_{ji} + \sum_{j=i+1}^{m} achiev.\, a_{ij} + \sum_{j=1}^{i-1} achiev.\, a_{ji})$
Total cost of developing and using an SoS	$TC = \sum_j^n \sum_i^m A_{ij}\mathrm{FIF}_i^{Ss} + \sum_k^n \sum_i^m A_{ii}\mathrm{FOP}_i^{Ss}$	25	Proprietary
Parameters for controlling the netcentric performance factor • Increment per interface • Penalty inc for unachievable • Penalty decrement for achievable i/f	Epsilon ϵ Penup Pendn	26	0.0015 1 1
Parameters for controlling the GA: • Mutation Rate • Number in Population • Number of Generations	Delta p g	27	0.003 40 50

4.1.6 How To Use the Method on Global Air Traffic Management. Global Air Traffic Management (GATM) is one of the largest SoS problems in existence. NextGen is a concept for modernizing air traffic control (ATC) in the United States to improve efficiency and reliability of control, and therefore the safety, of air travel, even in the face of more crowded skies in the future. Reducing delays, allowing more direct routing to improve fuel burn (for both efficiency and the environment), and reducing separation standards to allow more aircraft in the same space, while improving safety are the top level goals of NextGen (Federal Aviation Administration 2014). NextGen consists of 6 major POR level programs:

> "Automatic Dependent Surveillance-Broadcast (ADS-B) is FAA's satellite-based successor to radar. ADS-B makes use of GPS technology to determine and share precise aircraft location information, and streams additional flight information to the cockpits of properly equipped aircraft.
>
> Collaborative Air Traffic Management Technologies (CATMT) is a suite of enhancements to the decision-support and data-sharing tools used by air traffic management personnel. These enhancements will enable a more collaborative environment among controllers and operators, improving efficiency in the National Airspace System.
>
> Data Communications (Data Comm) will enable controllers to send digital instructions and clearances to pilots. Precise visual messages that appear on a cockpit display can interact with an aircraft's flight computer. Offering reduced opportunities for error, Data Comm will

supplant voice communications as the primary means of communication between controllers and flight crews.

National Airspace System Voice System (NVS) will supplant FAA's aging analog voice communication system with state-of-the-art digital technology. NVS will standardize the voice communication infrastructure among FAA facilities, and provide greater flexibility to the air traffic control system.

NextGen Weather will help reduce weather impact by producing and delivering tailored aviation weather products via SWIM, helping controllers and operators develop reliable flight plans, make better decisions, and improve on-time performance. NextGen Weather is accomplished through collaboration between FAA, NOAA and NASA.

System Wide Information Management (SWIM) is the network structure that will carry NextGen digital information. SWIM will enable cost-effective, real-time data exchange and sharing among users of the National Airspace System" (Federal Aviation Administration 2015).

Although NextGen is the US plan for ATC upgrades, European airspace is even more crowded and has additional issues due to the numerous sovereign national systems. Single European Sky (SES) is their master plan for ATM (EUROCONTROL - The European Organisation for the Safety of Air Navigation 2015), and SESAR (Single European Sky ATM Research) is the technology and systems portion of their ATC upgrade plans (European Commission of Transport 2015) (SESARJU 2015). Curiously, their website's audio says that the environment is the top goal, but the written materials

cite safety, efficiency, predictability and reduced cost for providing air traffic management (ATM) (what they call ATC) as the top goals, not bothering to cite environment at all.

In the Pacific region, which actually has slightly more passenger-miles flown than either North America or Europe, the current plan for improving ATM is under the auspices of the International Civil Aviation Organization (ICAO), embodied in the Asia/Pacific Seamless ATM Plan (Group, Asia/Pacific Seamless ATM Planning 2013). The number one, key attribute in that document is interoperability between different ATM regions, followed by safety, seamlessness of flight services between different regions, then efficiency. A secondary concern is simultaneity of changes in service modes among the regions. The Asia/Pacific region treats its member systems more like an acknowledged SoS than NextGen and SESAR. NextGen is slightly more in the strong central authority end of the spectrum of control, due to the principal participation of the federal government in ATC and tightly controlled certification for flight processes in the US. SESAR is about midway between Asia/Pacific and US in degree of central control for an SoS.

These details illustrate the necessity of reaching agreement on what to call the appropriate key attributes, along with careful definition of their meaning and how to measure them among all the stakeholders. Additional discussion of issues with NextGen is available in Haimes & Anderegg (Haimes and Anderegg 2015). This top-level agreement is key to making any analysis or recommendation useful to the broad group of stakeholders. This is especially true when close collaboration is necessary to achieve one of the key goals, such as the extremely technical goal of improved safety. If there is

anyone not following the agreed upon rules, safety inevitably suffers. When tackling this sort of global issue, it is extremely difficult to find the common ground without over-simplifying some issues; also difficult to agree on attribute definitions among all the competing voices in the discussion. Frequently the lofty goals suggesting themselves at first blush on seeing a problem get cut back in the hope of keeping all the member systems 'in the fold.' Establishing the dictionary and achieving stakeholder 'buy in' would have to be the initial priority for GATM. It would be very difficult in this arena. Nevertheless, harmonizing and improving GATM is a greatly to be desired, overarching goal, which everyone supports to some extent. That extent is largely determined by the affordability of the improvements. That is why affordability seems always to be a key attribute in evaluating the SoS.

Another problem for using the FILA-SoS approach on global ATM is that the structure of the participation is not at a peer to peer level, as it was in all previous examples of acknowledged SoS. There is a much more hierarchical nature to the systems' organization within ATM. Getting an airline company or a government to provide (or use) ATM services or capabilities in the desired way can stepwise add dozens or hundreds of aircraft (or flights – we still have to decide what the unit of system measure should be), or large coverage areas to the SoS, while the general aviation sector will require tedious, individual system co-option into the SoS. The FILA-SoS approach might have to be substantially modified to handle a hierarchical organization of systems, where some interfaces are still peer to peer, and others are up or down a hierarchical tree structure.

Selecting how to partition the numerous possible systems, as well as how to enumerate the possible capability elements, will also be a significant challenge for the goal of upgrading ATM globally. Civil air includes airlines of many sizes with many types of equipment, both fixed and rotary wing, as well as general aviation with many types and vintages of aircraft. There is also military and other government aircraft to consider. Ground facilities include airports, passenger and cargo facilities, maintenance (daily: such as fueling, minor inspection, remove & replace, and major: such as modifications and overhaul) as well as terminal and en route ATM control facilities. Space systems such as GPS, Inmarsat, or Iridium satellites may also have to be included. Enumerating and partitioning the classes of systems and capabilities would certainly need to be iterated with attribute selection and evaluation modeling approaches to be effective.

The various communications systems that provide the links between components will fit nicely into the achievable/unachievable interface formulation. Upgrading capabilities to include new digital radios would work well with the FILA-SoS negotiations framework and the time component of the development of capabilities over multiple epochs. So there are some factors about the global ATM problem that would fit quite well with the FILA-SoS approach, even if some parts present great difficulties.

There are large, detailed analyses underway by each ATM region on how to maintain safety while making the changes needed to upgrade the infrastructure of ATM systems, and how to implement changes piecemeal in the many, many components of the SoS. Safety as one of the primary goals is a difficult attribute to model. It is absolutely not considered adequate to have a ‘rough’ model of safety (one of the key thrusts of the FILA-SoS approach), but instead to require the most detailed and accurate modeling

possible to be able to prove any changes will be as safe as the existing systems. If the FILA-SoS approach is used to analyze policies, efficiency, timetables, or rough costs, it might do well. Attempting to use or even create 'rough' models of safety as an attribute would be very likely to discredit the approach entirely. However, the FILA-SoS approach is modular, so if a large scale, validated model could be made to work with the other components, one could theoretically use it within a FILA-SoS type approach.

4.2 RESULTS OF IMPLEMENTING THE METHOD

In the discussion of the results in this section, the definitions of the eight sub-graphs presented in Table 3.4 for each run during the GA have been changed as follows:

- The third graph on the top now has both the performance and flexibility attributes plotted in different colors to make room for the heat map
- The fourth graph now has a 'heat map' presentation of the frequency of ones in each chromosome position for the better performing half of the population.
- The penalty graph was removed to make room for the best chromosome graph of the current generation
- The last graph now has the best chromosome of the current generations' population plotted in the color coded upper triangular form

Later examples sometimes have a slightly different form of display because that subroutine was changed to be able to handle varying numbers of attributes and MFs, controlled entirely by the input data, using the same code for all types of problems.

4.2.1 Sensitivity Analysis. The ranges of items that were varied for sensitivity analysis included the following:

- The value of the netcentric performance increment (epsilon); from about 0.1% to 2% per achievable interface; ratio of penalty to reward was also varied through a range of 0.3 to 3.
- Changing epsilon requires adjustments to penup and pendn, as well as membership function limits to account for changes to average performance of SoS architectures, as well as the robustness limits because they correlate moderately with performance.
- Cost and performance inputs for various system elements, over a range of about 0.5 to 2 for the ratio of changes of key systems contributors.
- Protecting prior negotiated systems during mutation in the GA, to model a succeeding epoch of the wave model of SoS development where some systems with their interfaces had already negotiated their inclusion in the SoS and were not open to random selection. These are shown in pages E1-E11 of Appendix E, Supplementary Figures.
- Mutation rate was varied between 0.5% and 10% with no noticeable impact. The population size was varied from 20 to 5,000.
- The number of generations in the GA was varied from 20 to 500.
- Minor rule changes in the way attributes values contributed to SoS assessment were also varied over the course of the research.

In addition, coefficients of correlation between the number of systems, number of interfaces, all attribute evaluations, and the SoS assessment were run for each example. None of the variations made significant changes to the overall pattern of results, although convergence rate of the GA was occasionally different.

4.2.2 Results of Gulf War ISR Modeling. Representative generational snapshots in a GA run of 50 generations is shown in Figure 4.12 and Figure 4.13 for the first ISR model. All population graphs have been sorted by the overall SoS fitness, which is shown in the second graph in the top row of each snapshot. One can see the gradual improvement of the SoS fitness for the whole population from generation to generation in the four snapshots. The first generation still has the distribution forced from a few to many ones, but when sorted by the fitness, the correlation to chromosome number within the population is lost. In subsequent generations distribution of the number of ones is not forced, but governed by the mutation rate about the better chromosomes (with a few 'sports' from lower in the sort) that were selected for propagation to the next generation

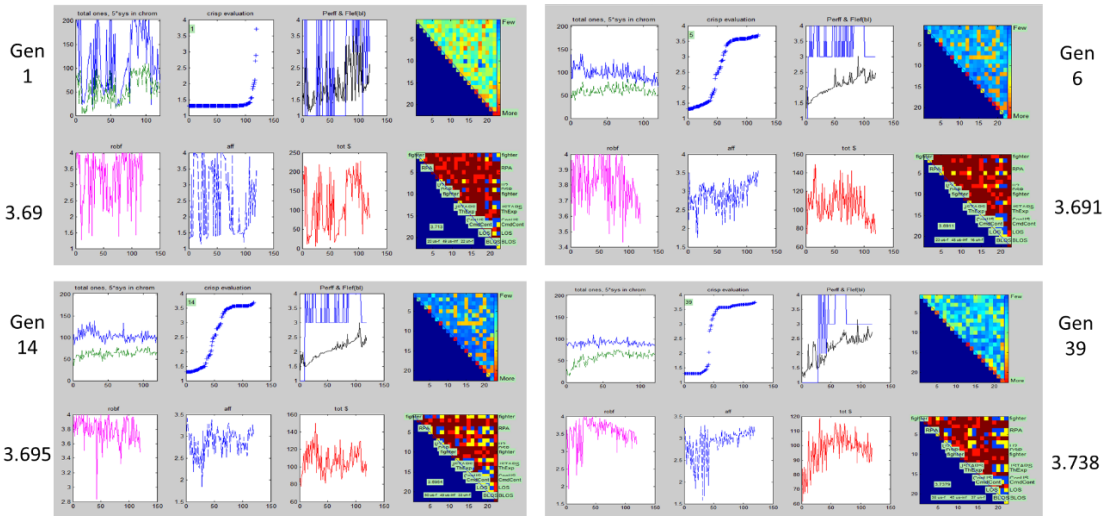


Figure 4.12. Intermediate progress through GA generations showing SoS fitness improvement

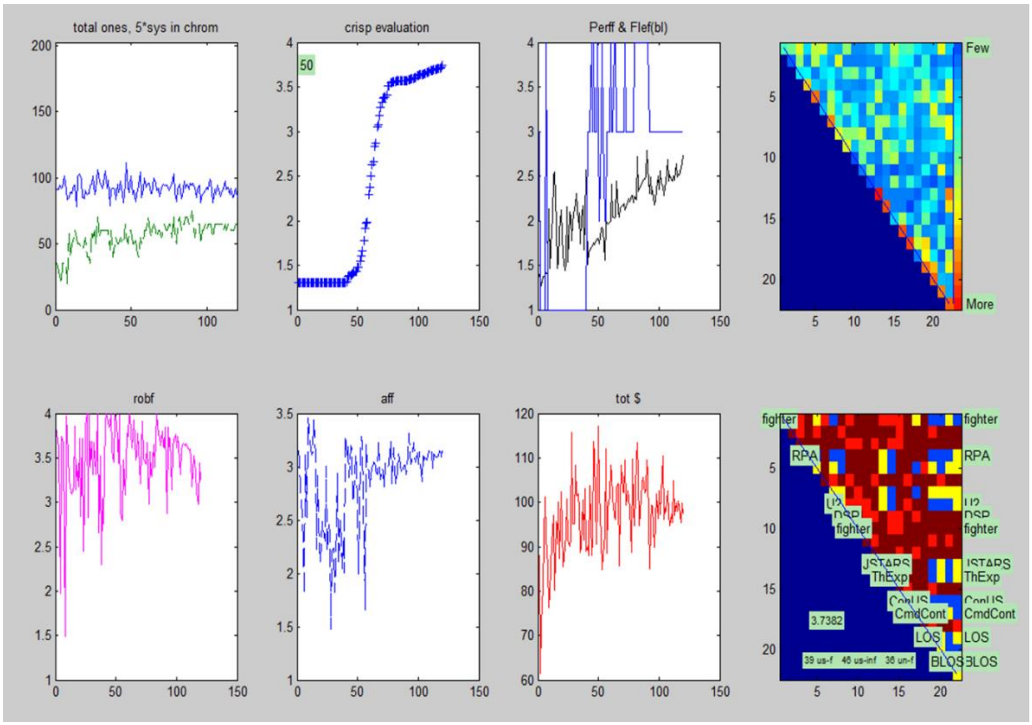


Figure 4.13. Typical 50th generation output graphs for GA of the ISR

Some GA optimization runs were made with relatively large populations and many generations. A population size of 300, is shown in Figure 4.14. The convergence plot in Figure 4.15 shows an improvement at generation 150 of 200. Most runs had no improvement after about 20 to 30 generations. A few still had some improvement as late as generation 70, but that was quite rare. The green line at the bottom of Figure 4.15 shows the assessment of the chromosome at the 20th percentile of overall fitness within the population in that generation. This example has relatively few top performing chromosomes, with about 30% of the population in a plateau at about 98% of the best value, as shown in the second subgraph on the top row of Figure 4.14. The best chromosome after this 200 generation run is shown in Figure 4.16.

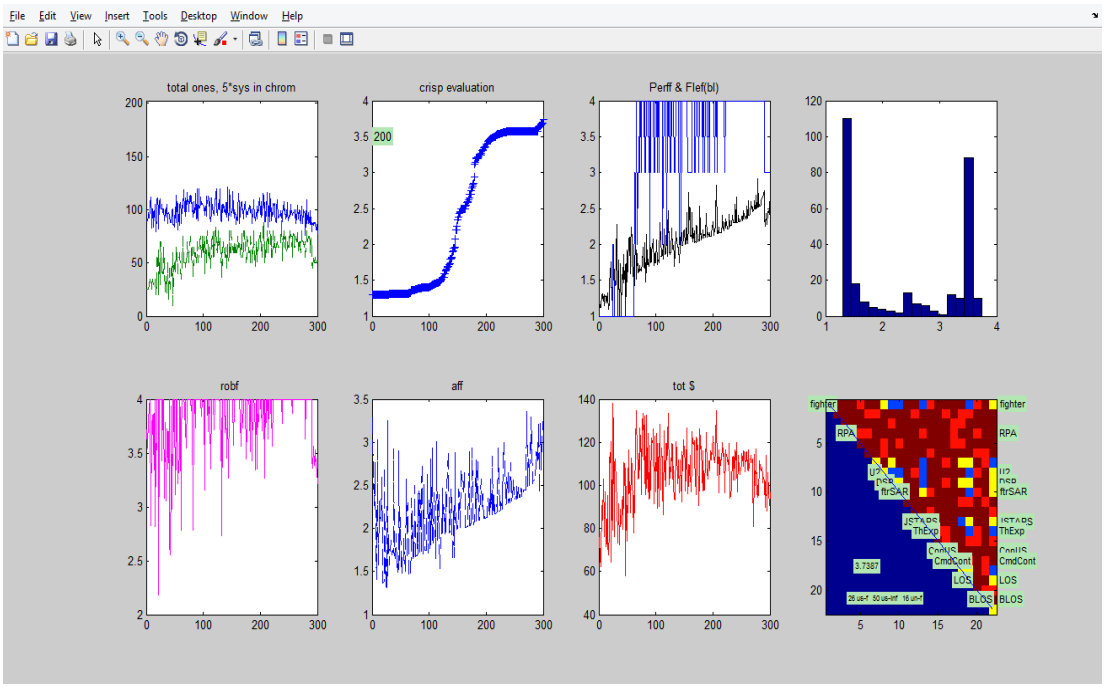


Figure 4.14. An ISR run of 200 gens with 300 in population

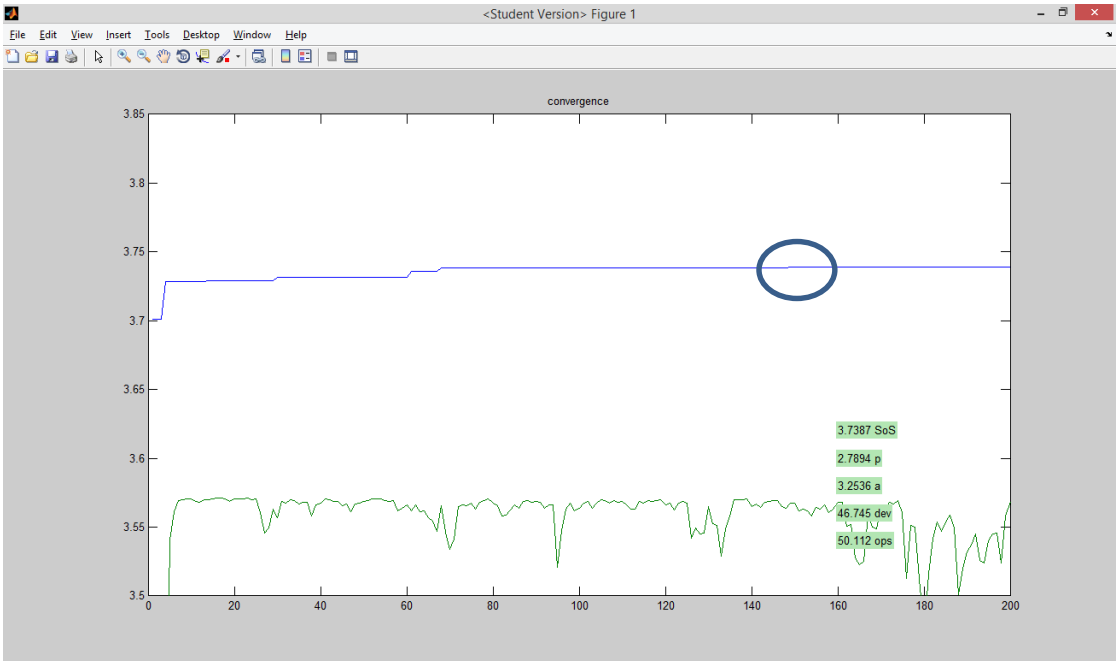


Figure 4.15. This convergence plot shows an ISR assessment still improving at generation 150

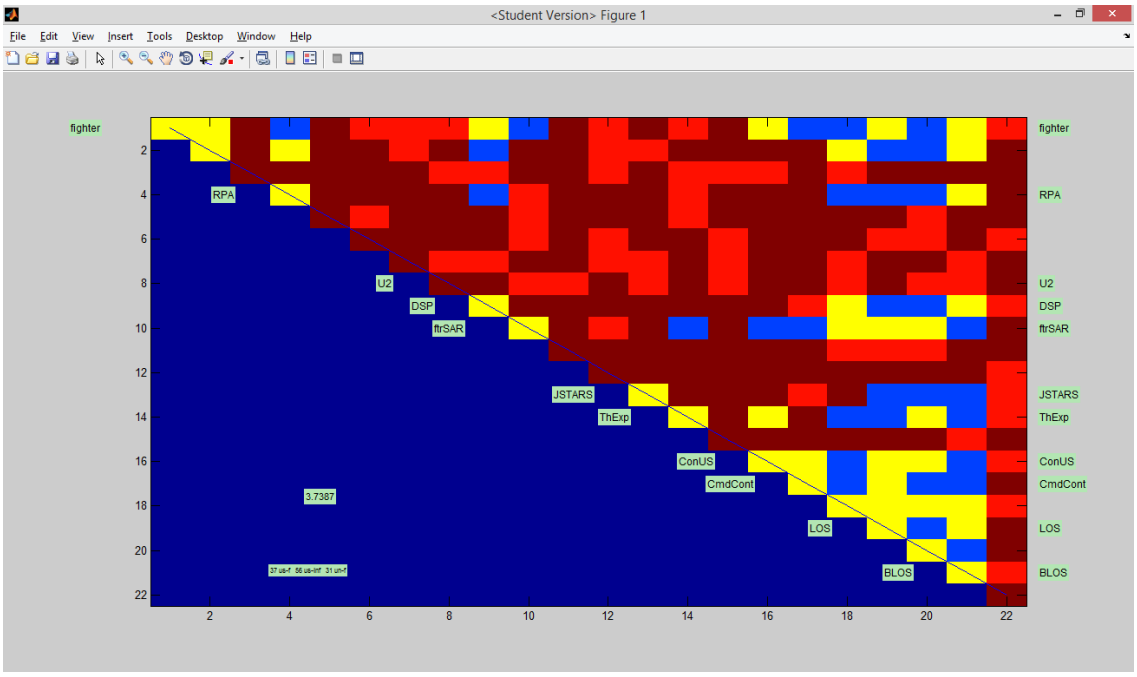


Figure 4.16. Final ISR SoS chromosome display for 200 generations

4.2.3 Results of OOTW Scenario Model. Figure 4.17 shows the result of a small valueExplore.m run to ensure that the MF edges are set in reasonable areas for each attribute, and that even with only a few chromosomes in this example, there are some acceptable SoS assessments.

The correlation coefficients between all the variables for this exploration run are shown in Table 4.17. Correlations between the position in the population, the number of systems and interfaces, the overall SoS assessment, each attribute evaluation, and the penalty for unachievable interfaces (I/Fs in the table) are shown. The relatively small correlation of each of these variables to the overall SoS assessment means that the SoS assessment does not weight any element more heavily than it should. Most attribute evaluations are not significantly cross-correlated, either. The highest correlation of any attribute evaluation with SoS assessment is only 26%. This means that the fuzzy assessor

is correctly picking architectures that satisfy several of the desires simultaneously, as intended.

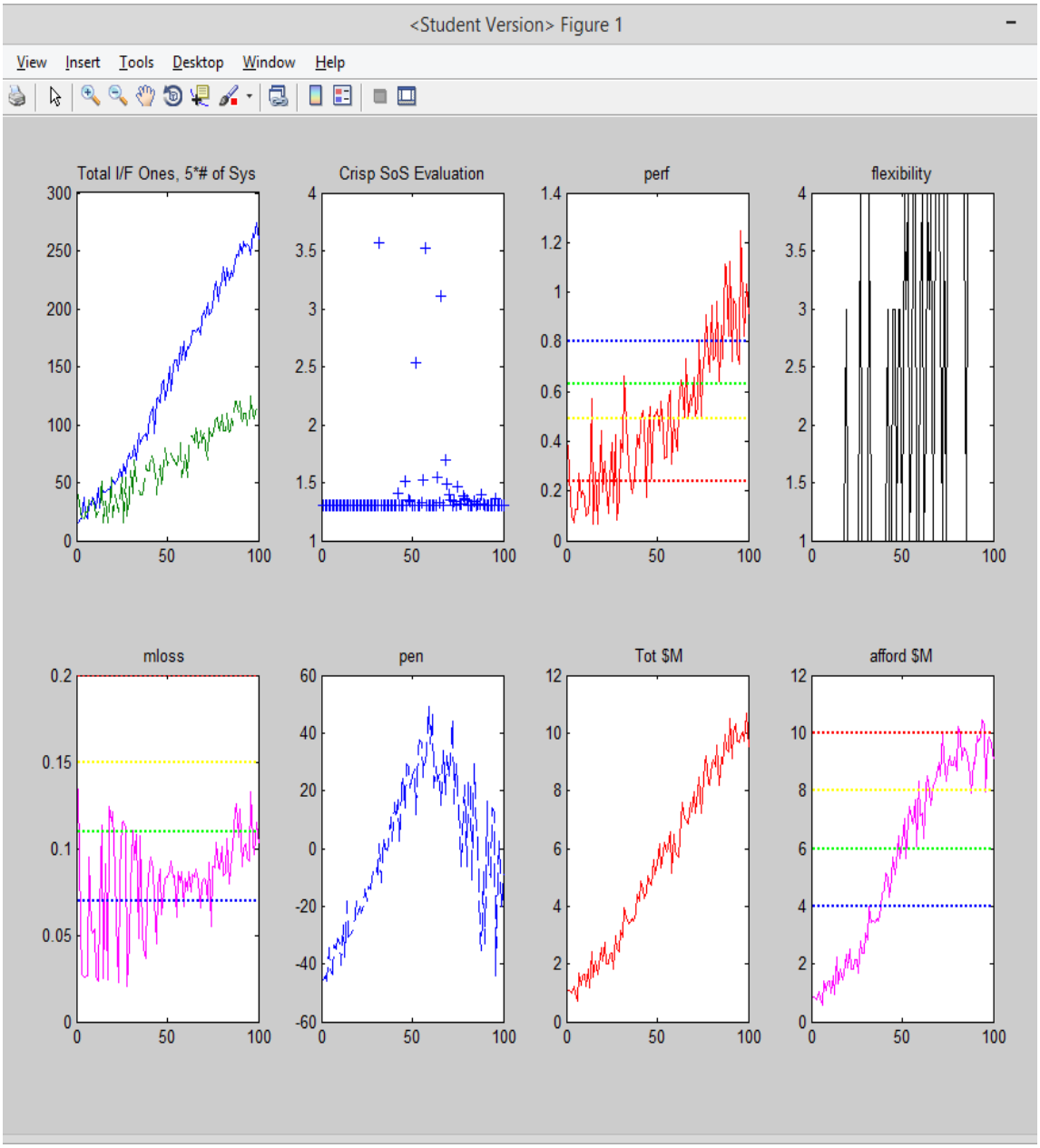


Figure 4.17. Biased number of ones in a small population explores the space adequately

Table 4.17. Correlation coefficients among all the OOTW attribute variables

	Pop #	Sum I/F	SoS Assess	Perf	Flex	Robust	Sum Sys	Penalty	Total $	Afford-ability
Pop #	1.000	0.9936	0.0331	0.8718	0.7502	0.4615	0.9347	0.4904	0.9874	0.9691
Sum I/F		1.0000	0.0327	0.8718	0.7582	0.4453	0.9338	0.4816	0.9921	0.9756
SoS Assess			1.0000	0.1324	0.2658	0.0625	0.1340	0.1483	0.0490	0.0699
Performance				1.0000	0.7719	0.6656	0.9562	0.1847	0.9070	0.8217
Flexibility					1.0000	0.4600	0.7912	0.2999	0.7789	0.7513
Robustness						1.0000	0.5608	0.0496	0.4840	0.4113
Sum Systems							1.0000	0.3846	0.9584	0.9169
Penalty								1.0000	0.4573	0.6247
Total $									1.0000	0.9755
Afford-ability										1.0000

The first generation, a randomized population (by the number and placement of ones in the chromosomes), when the GA was run on the OOTW model, showed fewer relatively good chromosomes than the other models, but otherwise behaved very similarly to the others. When large numbers (>50) were used in the population, there tended to be faster and smoother convergence to the ultimate arrangement. Since the early version GA implementation kept the top 20% including the best one and three other 'stray' chromosomes to build the next generation, populations less than 20 could not be used. Forty was the smallest population used in this research. That allowed a minimum of four of the better chromosomes (aside from the best one) to be kept for replication, mutation, crossover and transposition. Populations of 80, 100, or 120 were frequently used; a few times 1000 or even 5000 members were used, such as in the population for the OOTW problem. Figure 4.18 consists of snapshots of the GA for the OOTW problem; Figure 4.19 shows convergence (blue line) takes about the same number of generations for the smaller population sizes, but not as smoothly, and does not reach quite

to the same level as the larger population examples (the bumpy green line is the 20th percentile chromosome down from the fittest in each generation).

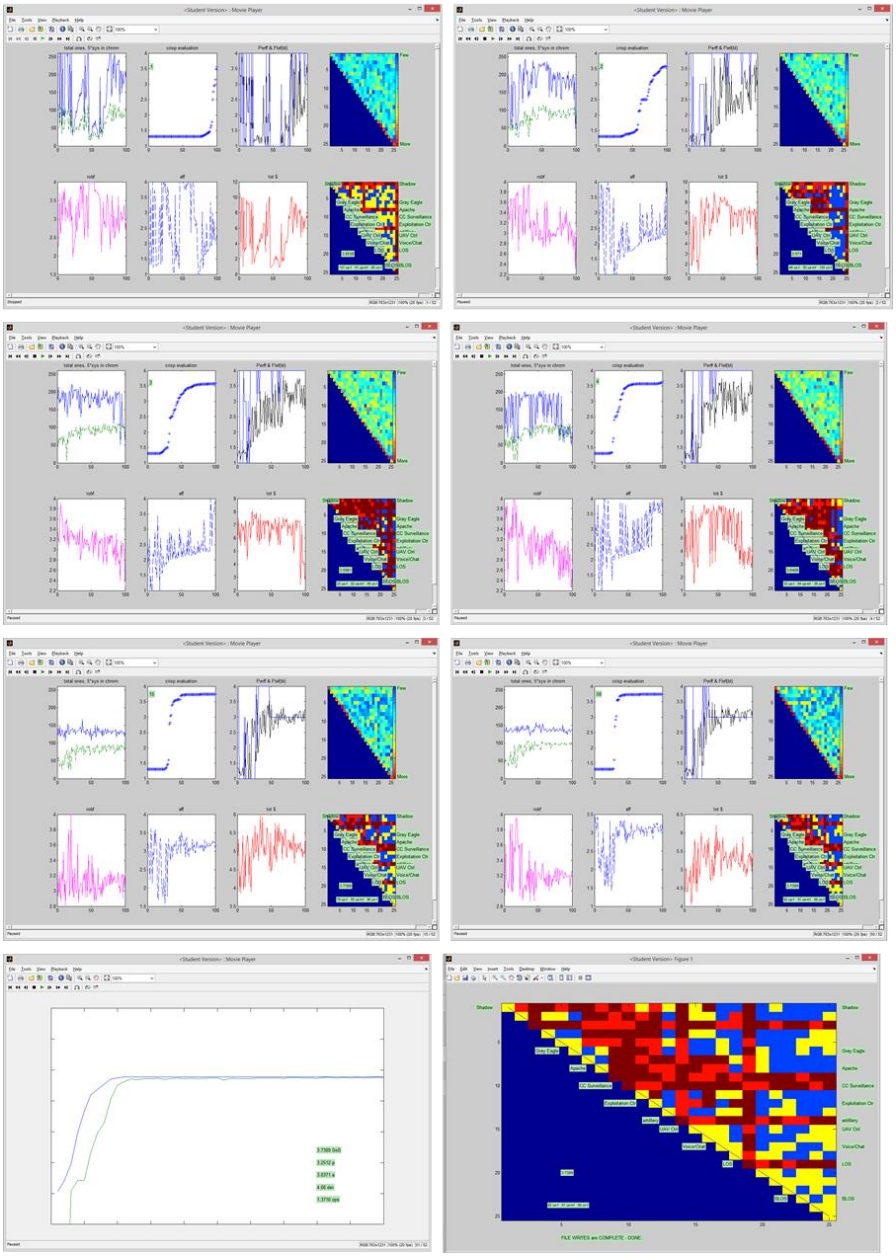


Figure 4.18. OOTW SoS GA snapshots with population =100, total generations = 50

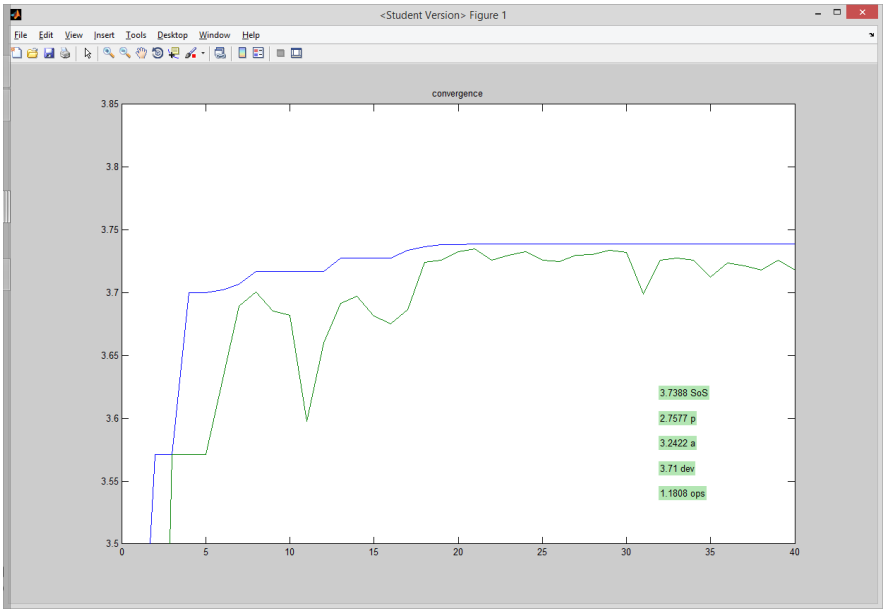


Figure 4.19. OOTW convergence with generations and population = 40

4.2.4 Results of SAR Modeling. The SAR model did not have exactly the same characteristics as the ISR model in the GA as shown in Figure 4.20 and Figure 4.21. There seemed to be a plateau of SoS assessments at the average level. The remainder of the evaluation functions operated quite similarly to the other SoS examples. With the most commonly used systems being the highest performing systems and the lowest cost systems. Figure 4.22 shows an example of implementing the second wave of SAR SoS development; it is not as good as the first wave because more systems joining in the second wave cost more, causing affordability to go down by a large amount.

The original inputs for the exploration portion of the method for SAR is shown in Figure 4.23; the result of changing the membership function ranges to get more attribute results into the 'above average' or 'acceptable' MF is shown in Figure 4.24. SoS assessment values on the left in the original do not exceed 2.6 on the scale of 4; but after easing the robustness MF limit (lower left graph), the example on the right has many

more population members around an SoS assessment of 3.6. The simple adjustment to the robustness MF mapping makes the SoS evaluation improve so much because it widens the choices available in the other attributes. In a real example, coordination among the stakeholders would be necessary to alter the membership function edges this way, but for demonstration purposes, it was only necessary to slightly alter the robustness MF edge to get different results in the valueExplore.m function, confirming that the GA could then be run far more successfully.

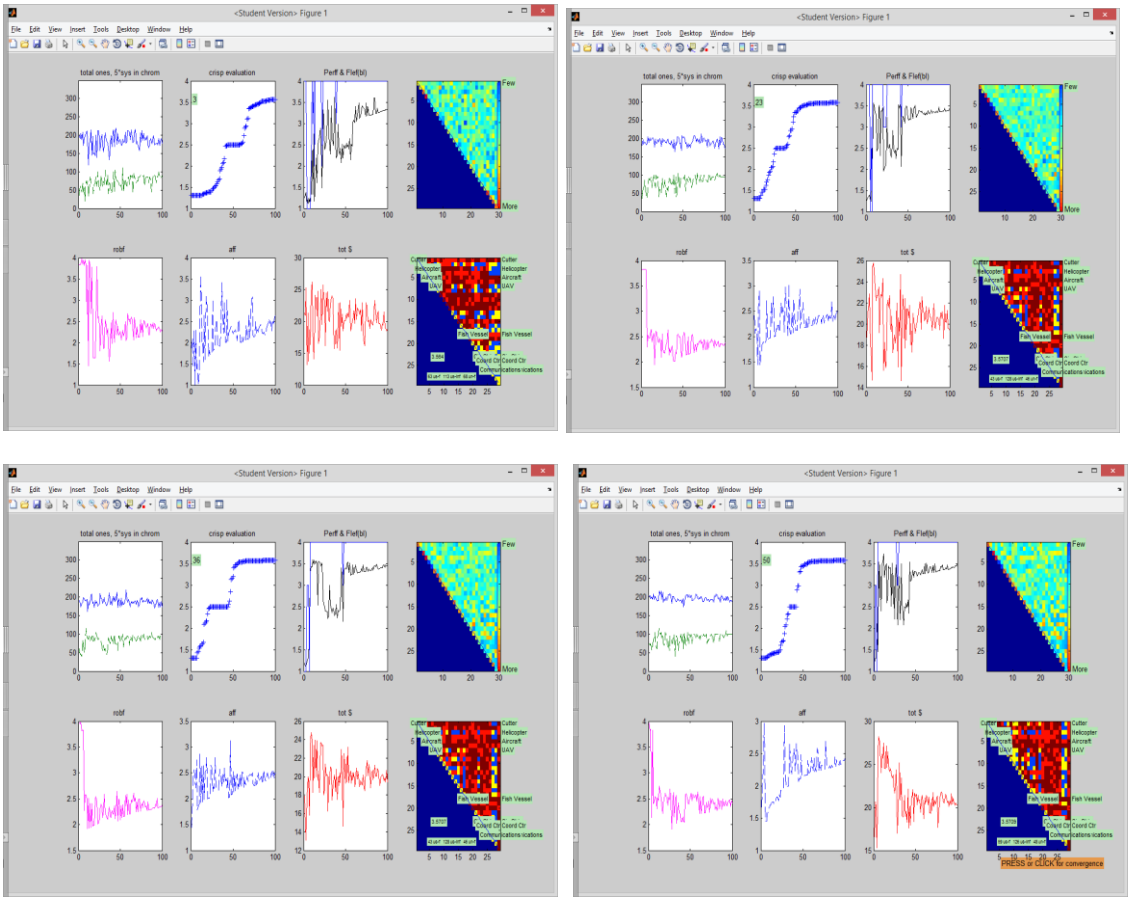


Figure 4.20. Snapshots of typical GA generations of 29 system SAR convergence

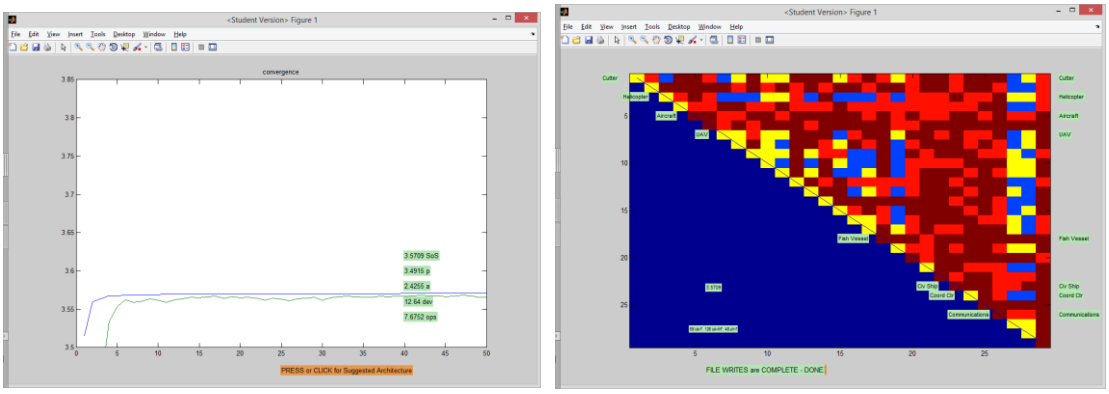


Figure 4.21. Convergence and final SAR SoS configuration, first wave epoch

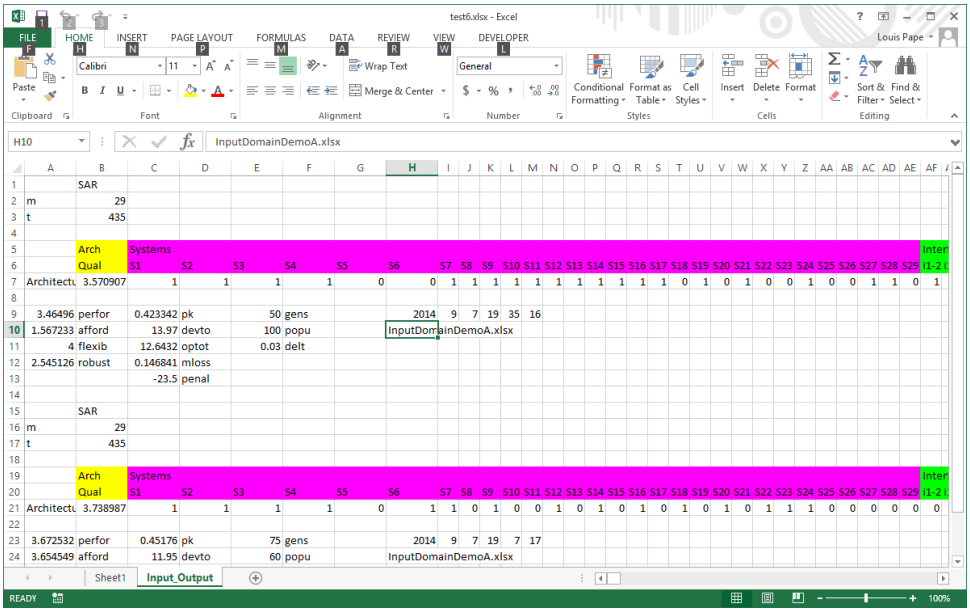


Figure 4.22. First wave on bottom; second wave on top

A different version of the fuzzy inference system with fewer rules and triangular membership functions was also used on the SAR problem. Figure 4.25 shows a selected chromosome very similar to that seen in Figure 4.20 and Figure 4.21 , with the original fuzzy inference system formulation. The alternate formulation appears to have slightly larger variations in the attribute evaluations per generation, but the plateaus in the SoS

assessments within a population are still there and the overall architecture suggestion is quite similar. In either formulation, improvement is not seen beyond approximately generation 20. More research is probably indicated to discover better ways to select appropriate MF shapes along with the crossover points between them for the attributes they characterize. This was merely a demonstration of the impact of choosing different MF shapes (trapezoidal vs. triangular in this case).

33	BUMP	0.01	0.4	1	penup	pendn	P,G,Mutat	60	40	0.02
34	Attributes		*mapfuzlow*	*1.5*	*2.5*	*3.5*	*4*			
35	Performance		0	0.09	0.18	0.32	0.5			
36	Affordability		-50	-38	-28	-18	-10			
37	Flexibility		0	1	2	3	4			
38	Robustness		-0.4	-0.3	-0.2	-0.11	0			

33	BUMP	0.01	0.4	1	penup	pendn	P,G,Mutat	60	40	0.02
34	Attributes		*mapfuzlow*	*1.5*	*2.5*	*3.5*	*4*			
35	Performance		0	0.09	0.18	0.32	0.5			
36	Affordability		-50	-38	-28	-18	-10			
37	Flexibility		0	1	2	3	4			
38	Robustness		-0.4	-0.35	-0.25	-0.15	0			

Figure 4.23. Robustness MF edges are changed between these two runs

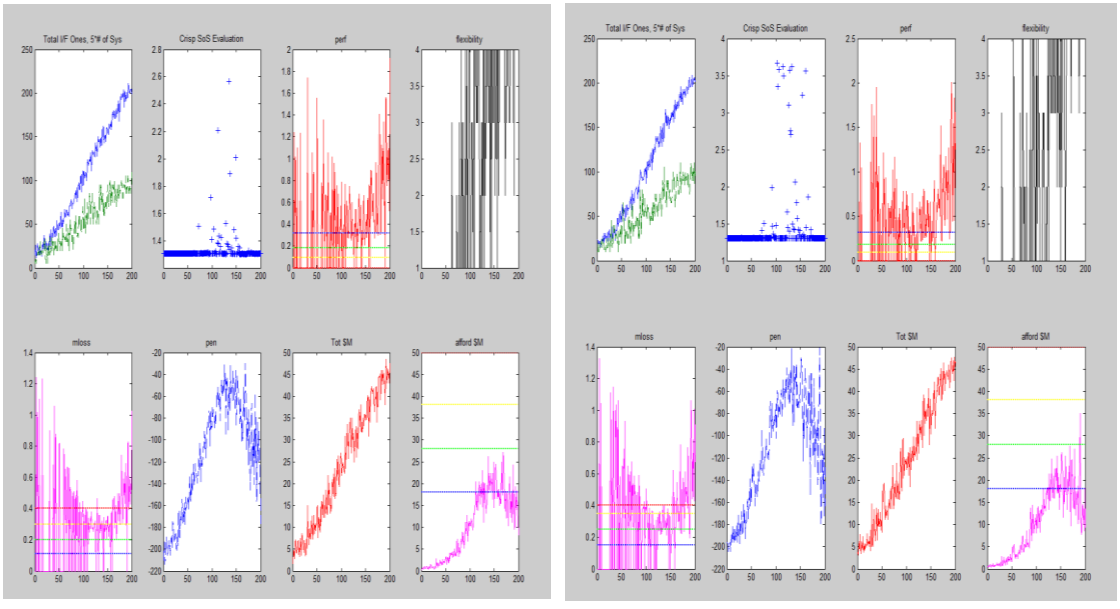


Figure 4.24. SAR runs show impact of robustness MF change in Figure 4.23

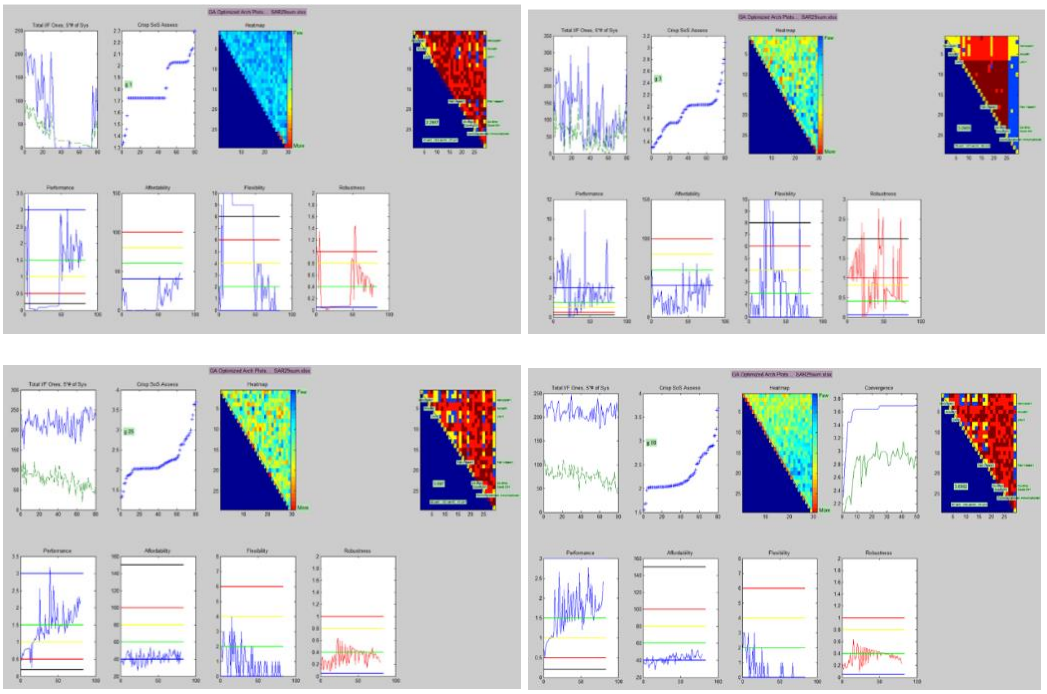


Figure 4.25. Alternate SAR formulation provides a similar architecture

4.2.5 Results of Toy Modeling. The toy model used functional dependency network analysis (FDNA) introduced by Pinto and Garvey to examine supply-chain problems (Garvey and Pinto 2009). It is a very different modeling paradigm than the ISR and SAR problems. Instead of a netcentric factor in the performance, evaluation, it uses a very complicated network application of strength of dependency (SOD) and criticality of dependency (COD) for each interface between nodes. The initial toy model was solved for choices of only one system from each type of system, with different SOD and COD values for each interface. Later, the FDNA problem was solved in general. Then, any number of each type of system was allowed in the SoS.

4.2.5.1 Initial Toy model results. The original implementation of the Missouri Toy problem admitted only a few choices; it required one of each of the five types of

systems; there were only choices for the 3 central systems: SatA, UAV, and SatB. There were only 8x6x6 = 288 choices possible. These were easy to exhaustively list and evaluate, as shown in Figure 4.26 to Figure 4.28. Here the chromosome has successive selections of SatA1 – SatA8, UAV0 – UAV5 and SatB1-SatB6, selected in a nested loop to run through all 288 chromosomes. The input SOD and COD of each component is shown on page E13 of Appendix E. The selection of different single systems provide different strength and criticality of dependency of the resulting five member SoS. The output at the Carrier is the green line on each of the graphs. When Ground has all its capability of 100 in Figure 4.26, there is no dependence on selection of intermediary systems in the result. When Ground capability is reduced, as in Figure 4.27 and Figure 4.28, then one can see there is impact to the result at the Carrier that is dependent on selected path, frequently showing large changes for a single system's different choice.

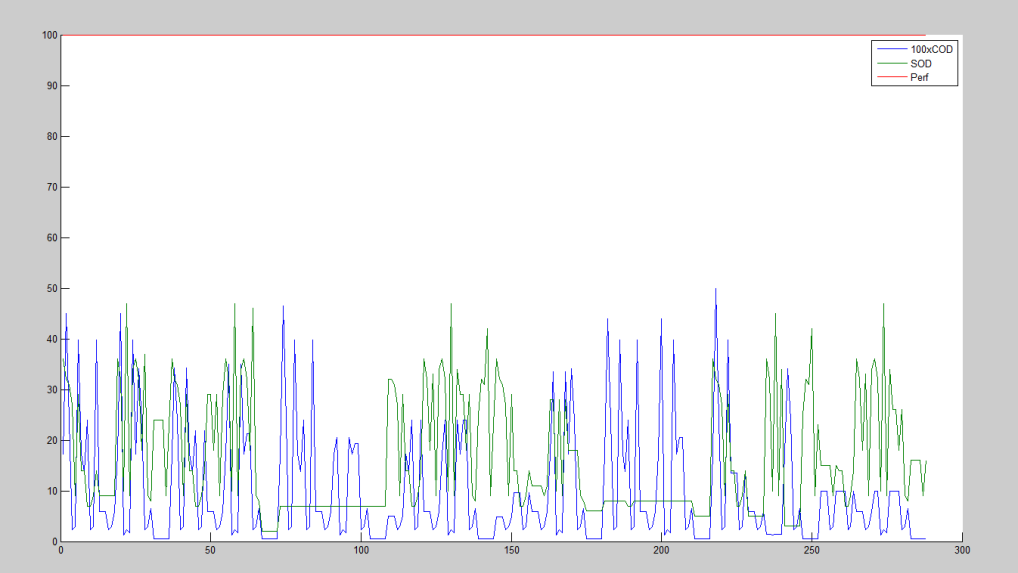


Figure 4.26. Output performance for Ground station input performance of 100

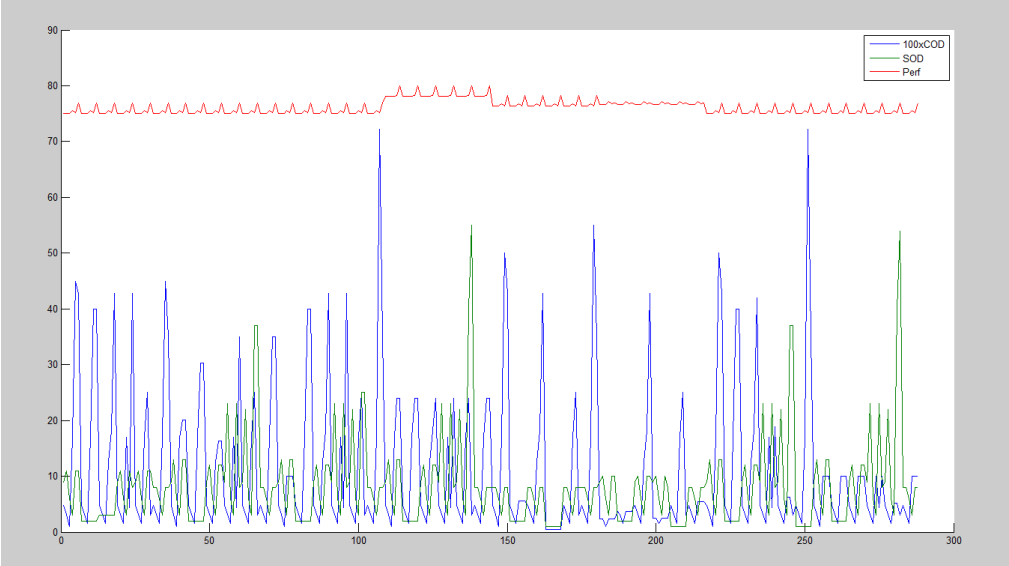


Figure 4.27. Output performance for Ground station input performance of 75

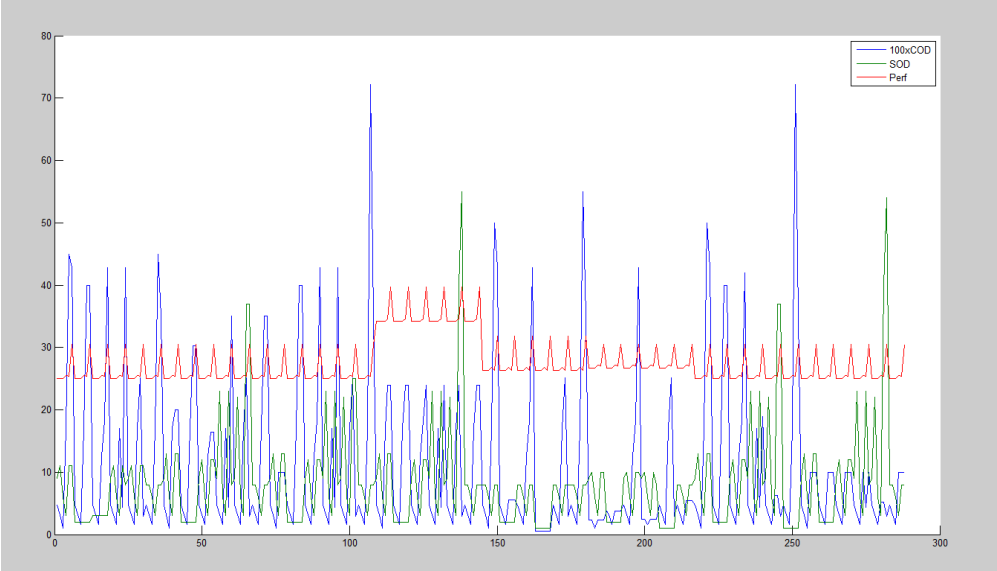
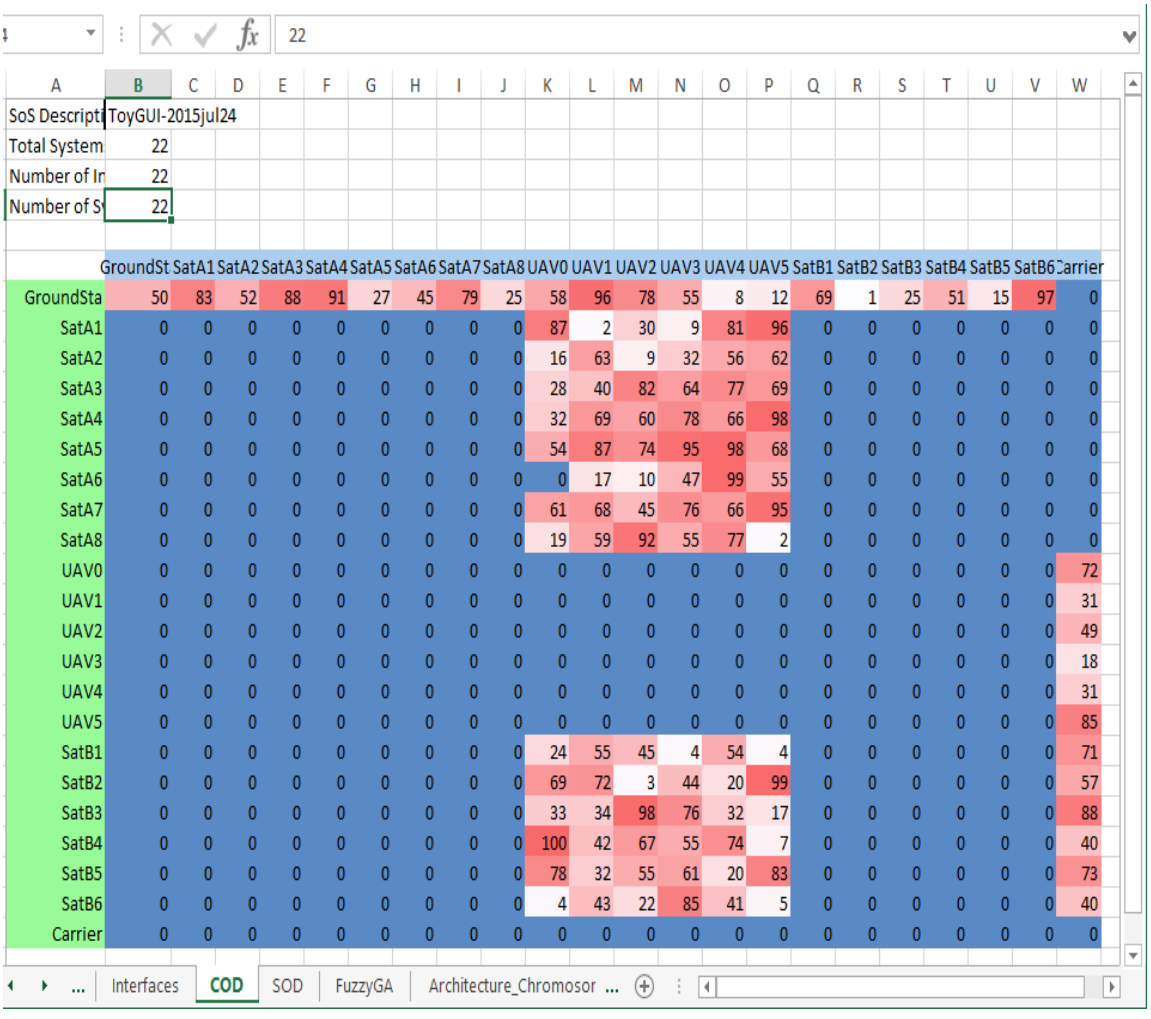


Figure 4.28. Output performance for Ground station input performance of 25

4.2.5.2 Generalized FDNA implementation results. With the change to allow any number of each of the three central system types in the Toy problem, it now looks much more like the other SoS problems, with numerous potential systems and interfaces.

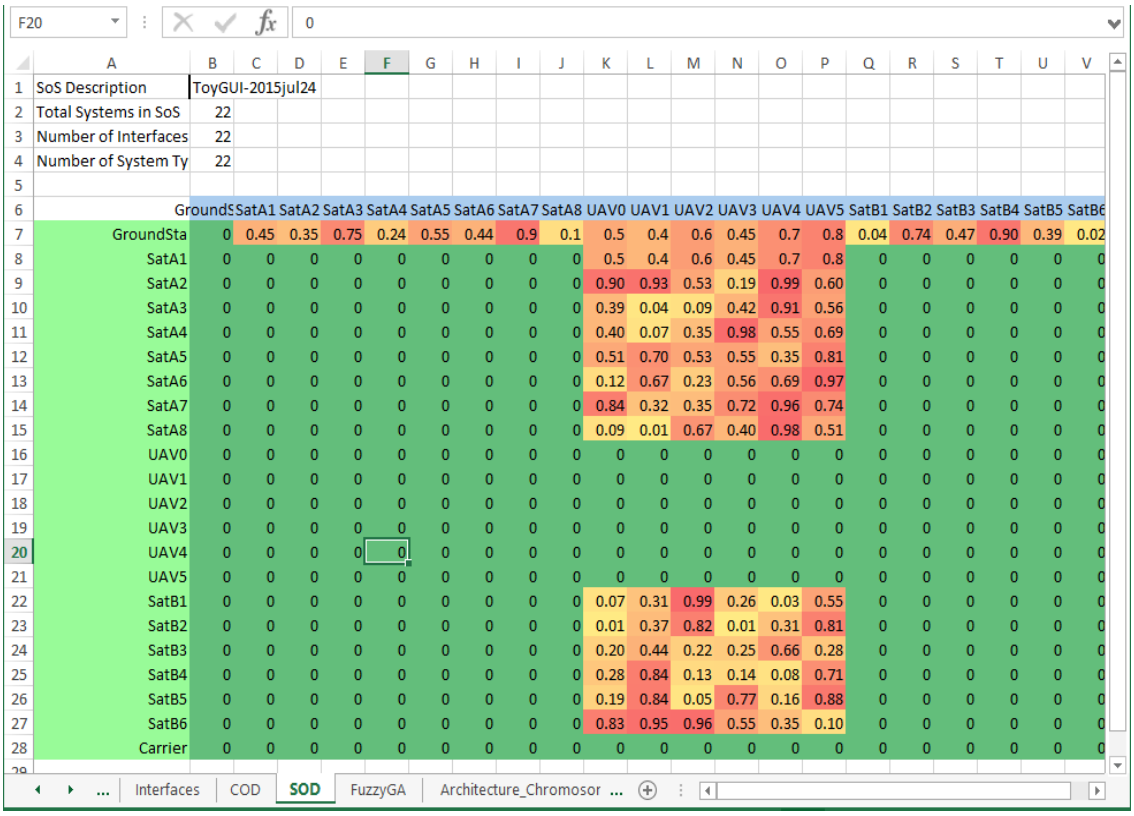
Neither the concept of netcentricity nor achievability of interfaces apply to the Toy problem; all the interfaces are possible to the chromosome, but only some have SOD and COD values that provide connectivity in the correct places for the problem. Interfaces with no corresponding SOD/COD values are ignored. Allowed SOD and COD values were filled with random numbers in the appropriate ranges, as shown in Figure 4.29 and Figure 4.30 through color-coding.



SoS Descripti	ToyGUI-2015jul24	
Total System	22	
Number of In	22	
Number of S	22	

	GroundSt	SatA1	SatA2	SatA3	SatA4	SatA5	SatA6	SatA7	SatA8	UAV0	UAV1	UAV2	UAV3	UAV4	UAV5	SatB1	SatB2	SatB3	SatB4	SatB5	SatB6	Carrier
GroundSta	50	83	52	88	91	27	45	79	25	58	96	78	55	8	12	69	1	25	51	15	97	0
SatA1	0	0	0	0	0	0	0	0	0	87	2	30	9	81	96	0	0	0	0	0	0	0
SatA2	0	0	0	0	0	0	0	0	0	16	63	9	32	56	62	0	0	0	0	0	0	0
SatA3	0	0	0	0	0	0	0	0	0	28	40	82	64	77	69	0	0	0	0	0	0	0
SatA4	0	0	0	0	0	0	0	0	0	32	69	60	78	66	98	0	0	0	0	0	0	0
SatA5	0	0	0	0	0	0	0	0	0	54	87	74	95	98	68	0	0	0	0	0	0	0
SatA6	0	0	0	0	0	0	0	0	0	0	17	10	47	99	55	0	0	0	0	0	0	0
SatA7	0	0	0	0	0	0	0	0	0	61	68	45	76	66	95	0	0	0	0	0	0	0
SatA8	0	0	0	0	0	0	0	0	0	19	59	92	55	77	2	0	0	0	0	0	0	0
UAV0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	72
UAV1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31
UAV2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	49
UAV3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18
UAV4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	31
UAV5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	85
SatB1	0	0	0	0	0	0	0	0	0	24	55	45	4	54	4	0	0	0	0	0	0	71
SatB2	0	0	0	0	0	0	0	0	0	69	72	3	44	20	99	0	0	0	0	0	0	57
SatB3	0	0	0	0	0	0	0	0	0	33	34	98	76	32	17	0	0	0	0	0	0	88
SatB4	0	0	0	0	0	0	0	0	0	100	42	67	55	74	7	0	0	0	0	0	0	40
SatB5	0	0	0	0	0	0	0	0	0	78	32	55	61	20	83	0	0	0	0	0	0	73
SatB6	0	0	0	0	0	0	0	0	0	4	43	22	85	41	5	0	0	0	0	0	0	40
Carrier	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.29. COD values for generalized Toy problem



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	SoS Description	ToyGUI-2015jul24																				
2	Total Systems in SoS	22																				
3	Number of Interfaces	22																				
4	Number of System Ty	22																				
5																						
6		GroundS	SatA1	SatA2	SatA3	SatA4	SatA5	SatA6	SatA7	SatA8	UAV0	UAV1	UAV2	UAV3	UAV4	UAV5	SatB1	SatB2	SatB3	SatB4	SatB5	SatB6
7	GroundSta	0	0.45	0.35	0.75	0.24	0.55	0.44	0.9	0.1	0.5	0.4	0.6	0.45	0.7	0.8	0.04	0.74	0.47	0.90	0.39	0.02
8	SatA1	0	0	0	0	0	0	0	0	0	0.5	0.4	0.6	0.45	0.7	0.8	0	0	0	0	0	0
9	SatA2	0	0	0	0	0	0	0	0	0	0.90	0.93	0.53	0.19	0.99	0.60	0	0	0	0	0	0
10	SatA3	0	0	0	0	0	0	0	0	0	0.39	0.04	0.09	0.42	0.91	0.56	0	0	0	0	0	0
11	SatA4	0	0	0	0	0	0	0	0	0	0.40	0.07	0.35	0.98	0.55	0.69	0	0	0	0	0	0
12	SatA5	0	0	0	0	0	0	0	0	0	0.51	0.70	0.53	0.55	0.35	0.81	0	0	0	0	0	0
13	SatA6	0	0	0	0	0	0	0	0	0	0.12	0.67	0.23	0.56	0.69	0.97	0	0	0	0	0	0
14	SatA7	0	0	0	0	0	0	0	0	0	0.84	0.32	0.35	0.72	0.96	0.74	0	0	0	0	0	0
15	SatA8	0	0	0	0	0	0	0	0	0	0.09	0.01	0.67	0.40	0.98	0.51	0	0	0	0	0	0
16	UAV0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	UAV1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	UAV2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	UAV3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	UAV4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	UAV5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	SatB1	0	0	0	0	0	0	0	0	0	0.07	0.31	0.99	0.26	0.03	0.55	0	0	0	0	0	0
23	SatB2	0	0	0	0	0	0	0	0	0	0.01	0.37	0.82	0.01	0.31	0.81	0	0	0	0	0	0
24	SatB3	0	0	0	0	0	0	0	0	0	0.20	0.44	0.22	0.25	0.66	0.28	0	0	0	0	0	0
25	SatB4	0	0	0	0	0	0	0	0	0	0.28	0.84	0.13	0.14	0.08	0.71	0	0	0	0	0	0
26	SatB5	0	0	0	0	0	0	0	0	0	0.19	0.84	0.05	0.77	0.16	0.88	0	0	0	0	0	0
27	SatB6	0	0	0	0	0	0	0	0	0	0.83	0.95	0.96	0.55	0.35	0.10	0	0	0	0	0	0
28	Carrier	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.30. SOD values for generalized Toy problem

The exploration of the generalized, modified Missouri Toy problem architecture can now be seen in Figure 4.31. This shows how the random chromosomes fit within the attribute evaluation membership functions allowing for the SoS assessment to work well. Several snapshot views of the GA generations are shown next in Figure 4.32, and the final chromosome with convergence is shown in Figure 4.33, with much similarity to the previous SoS examples. The correlation coefficients between SoS assessment and attribute evaluations in the Toy problem in Table 4.18 are all less than 0.6 except affordability, which was artificially manipulated to control the selection of multiple systems. This would not be regarded as significant in most cases.

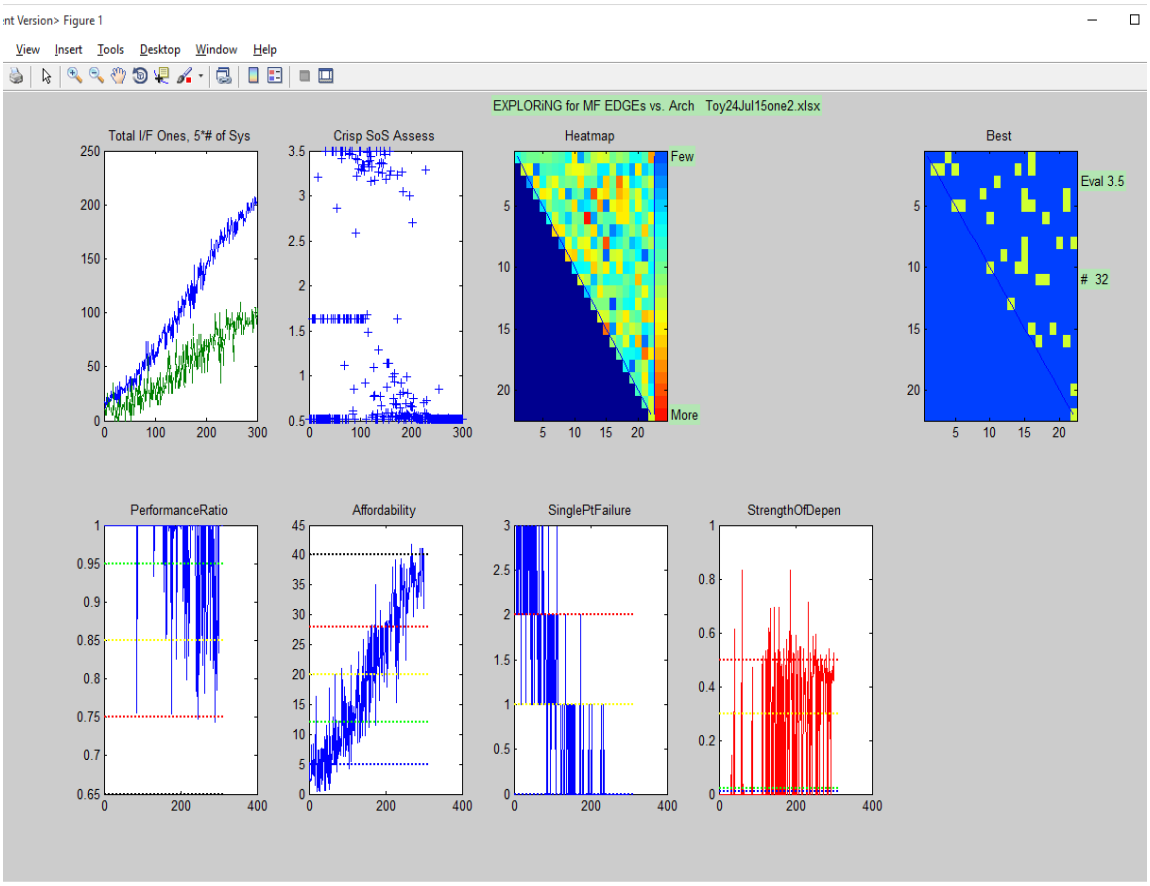


Figure 4.31. Exploration of the space with 300 biased Toy chromosomes

Table 4.18. Cross correlation matrix for the Toy problem shows minor correlation

q	i/f	5*sys	crisp	'PerfRatio'	'Afford'	'SingPtFailure'	'StrOfDepen'
1	0.99009	0.92912	-0.33646	-0.46849	-0.93025	0.82492	-0.57218
0.99009	1	0.92976	-0.37072	-0.46607	-0.93175	0.81132	-0.57316
0.92912	0.92976	1	-0.38345	-0.52782	-0.99021	0.85458	-0.60375
-0.33646	-0.37072	-0.38345	1	0.28594	0.38387	-0.066851	0.49082
-0.46849	-0.46607	-0.52782	0.28594	1	0.52006	-0.35869	0.46165
-0.93025	-0.93175	-0.99021	0.38387	0.52006	1	-0.85366	0.58332
0.82492	0.81132	0.85458	-0.066851	-0.35869	-0.85366	1	-0.51362
-0.57218	-0.57316	-0.60375	0.49082	0.46165	0.58332	-0.51362	1

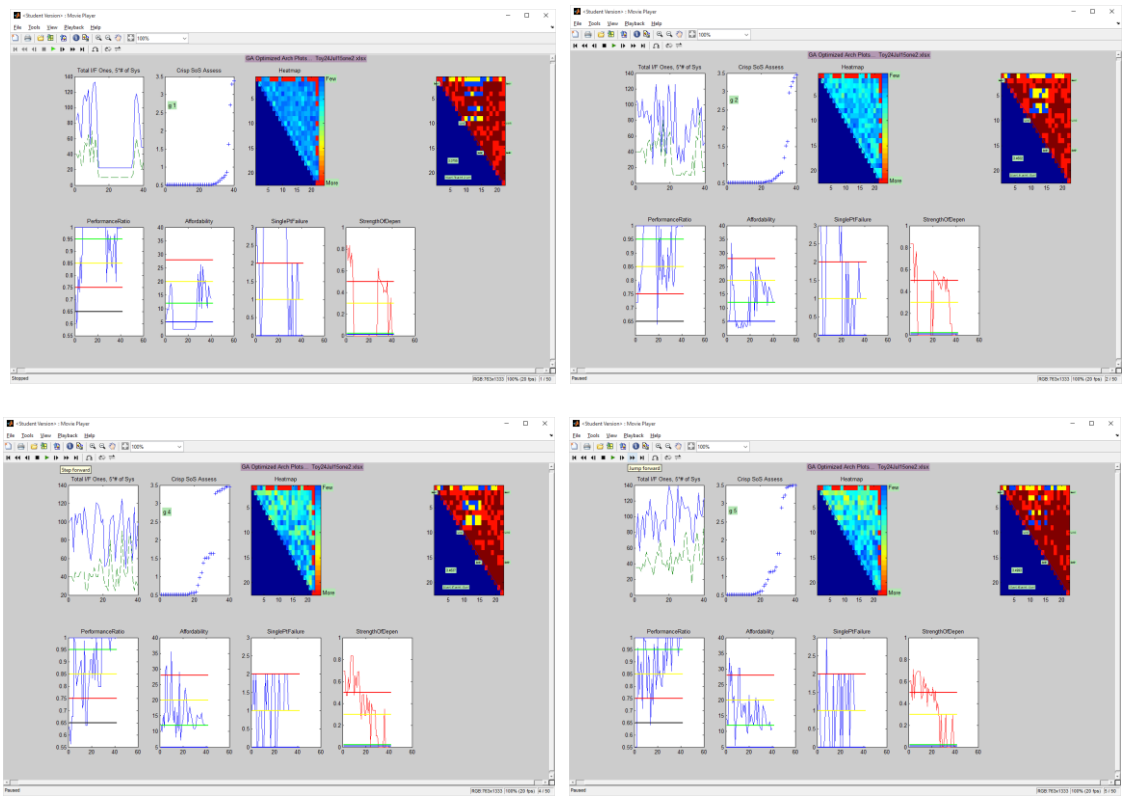


Figure 4.32. Early generations of Toy problem GA run shows selected interfaces changing

4.2.6 Validation with a Large, Real World Example. Propriety data from the Army and MITRE were used to validate the method with a large SoS problem. The architecture generation method seems to be capable of dealing with increased SoS size. The computational scalability of the method seems to be quite good. Matrices are used primarily for keeping track of the model data and relationships. No matrix inversions or large matrix multiplies are required that might cause a programming implementation to run out of memory. The fuzzy GA runs in a few seconds to a few minutes in Matlab on a high-end PC, depending on population size and number of generations. The most time consuming computational task is reading and writing to Excel spreadsheets within

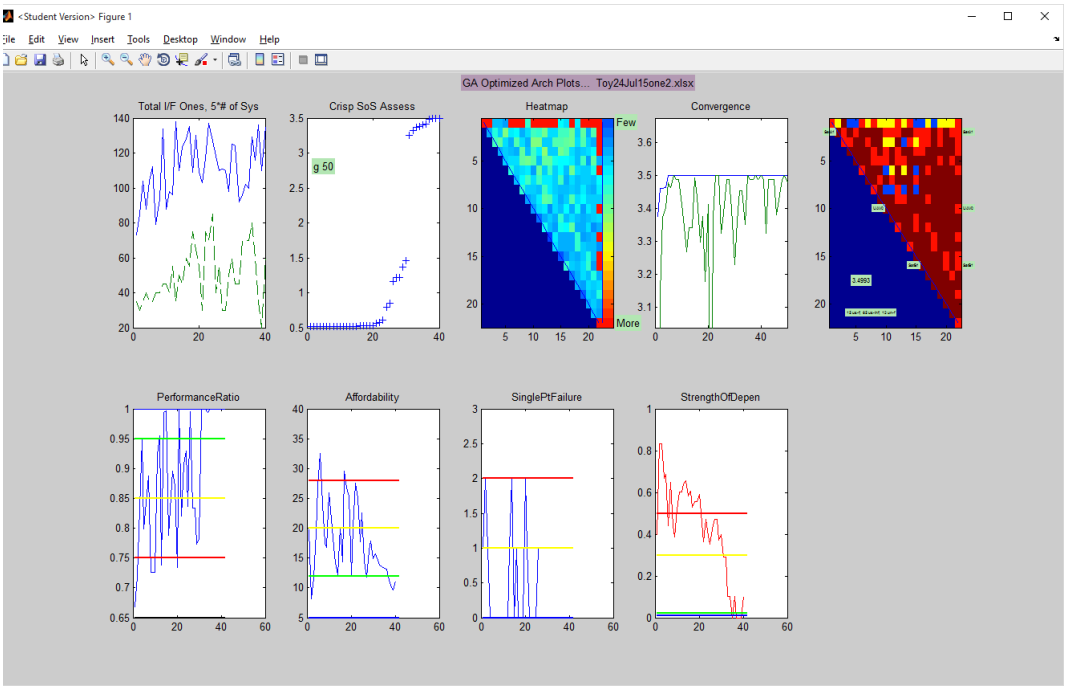


Figure 4.33. Final generation of Toy problem; convergence plateaued around generation seven of 50

Matlab for interoperability with the other segments of FILA-SoS. The validation problem from MITRE had 111 systems, with 74 capabilities as shown in Figure 4.34. The figure shows 111 systems with costs and performance (down the left) and 74 capabilities (across the top); shaded areas at intersections represent the capabilities of each system. It had seven KPAs with five levels of granularity in the membership functions. The fuzzy inference system had 18 rules, shown in Figure 4.35, compared to the 11 rules in the four attribute ISR, OOTW, SAR, and Toy problems. The SARone example with triangular MFs had only 4 rules. The principal and most time-consuming things that had to be changed in the software for the much larger validation problem were the display subroutines. These were successfully modified to be quite general now, as shown in Appendix B.

The approach and process steps in the domain independent portion of the method worked quite well. The most risky area for expansion is communication with and level setting among all the systems, capabilities and stakeholders in a larger SoS. Gathering the other domain dependent data such as cost and schedule estimates, deciding what minor changes could be made, deconstructing system capabilities, and developing attribute evaluation algorithms is more time-consuming for the greater number of systems. However, with that many systems, patterns emerge and many elements might be filled in rapidly.

Growth in the number of KPAs would significantly drive the amount of analysis required to create evaluation algorithms, choose membership function shapes for each one, and check their validity. Growth in the number of KPAs would very likely also drive a larger number of rules in the FAM. When all these model parameters grow in number, the number of iterations in the sampling runs can increase substantially to insure everything is correctly coded for all the combinations. The coordination and SME reviews grow with the number of systems. On the other hand, as the number of systems grows, the impact of individual systems is more diffuse; therefore, the need for every system to be modeled very accurately (as well as errors in modeling the impacts of each bit in the chromosome) diminishes. Therefore, there are several straightforward linear factors that increase the time and effort it takes to create, socialize, and vet the larger model, but the experience in the FILA-SoS validation problem shows this could be fairly reasonable if the stakeholder community cooperates.

The attributes definitions and the granularity of each attribute were provided by the customer in the validation problem. Once again, the population evaluations were not

strongly correlated, as shown in Table 4.19, although more highly correlated than the prior ‘made up’ examples in the Table 4.17 and Table 4.18. This means that the stakeholders had chosen good attributes for their problem. Finally, the upper triangular matrix form of the architecture as shown in Figure 4.36, is starting to show bands of **un**selected systems where they were relatively expensive, even though those systems could provide many capabilities; many of those capabilities were available from the other systems as well. This is a reasonable way to select an arrangement of systems and interfaces for an SoS

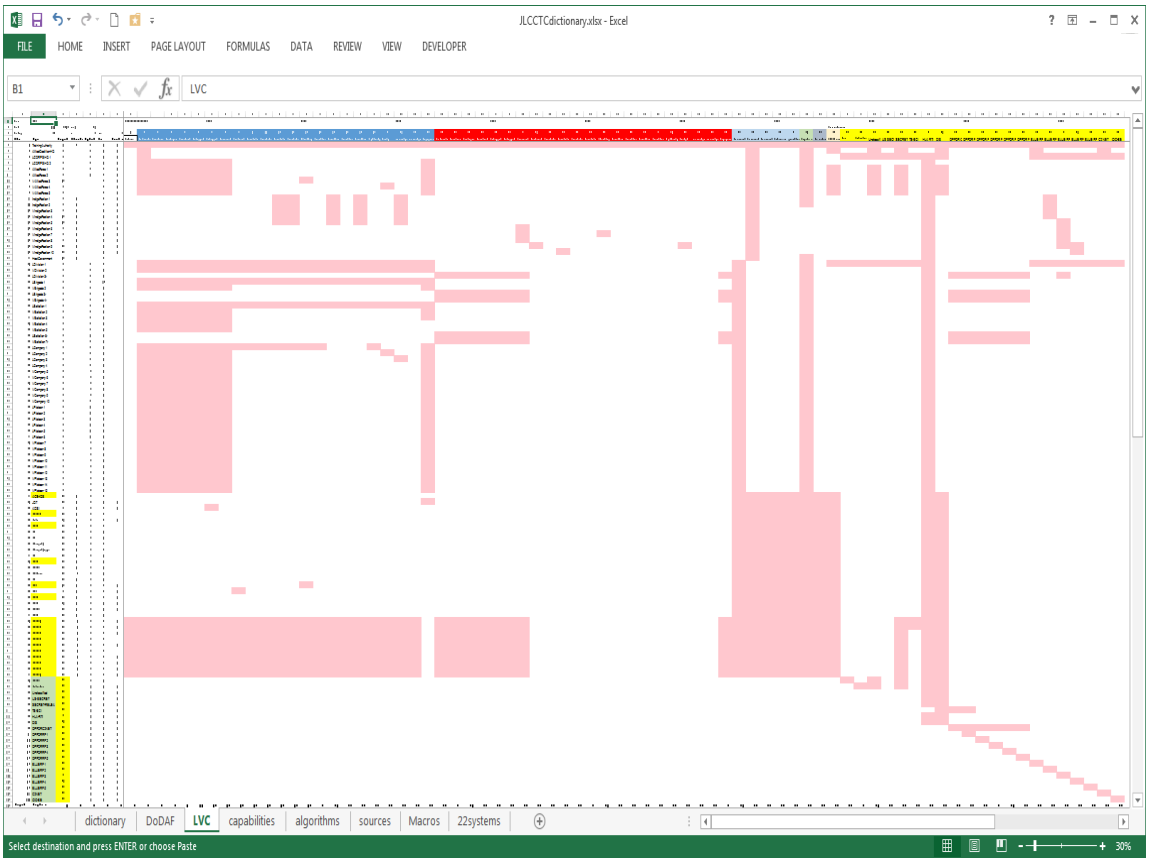


Figure 4.34. Very large input data matrix for the LVC problem (gray cells contain ‘1’)

Figure 4.35. 18 fuzzy rules with seven attributes on the left, compared to 11 rules and four attributes of other models

Table 4.19. Correlation coefficients for the LVC problem

	Pop Seq #	# of I/F	# 0f Sys	SoS Assess	Attr 1	Attr 2	Attr 3	Attr 4	Attr 5	Attr 6	Afford
Pop Seq #	1	0.997	0.984	0.637	0.935	0.997	0.861	0.536	0.98	0.67	-0.996
# of I/F		1	0.988	0.633	0.944	1	0.87	0.564	0.984	0.674	-0.999
# 0f Sys			1	0.622	0.946	0.988	0.862	0.569	0.994	0.683	-0.988
SoS Assess				1	0.514	0.634	0.692	0.657	0.62	0.589	-0.632
Attr 1					1	0.944	0.747	0.396	0.952	0.689	-0.945
Attr 2						1	0.87	0.564	0.984	0.674	-0.999
Attr 3							1	0.67	0.854	0.589	-0.871
Attr 4								1	0.559	0.312	-0.561
Attr 5									1	0.687	-0.985
Attr 6										1	-0.678
Afford											1

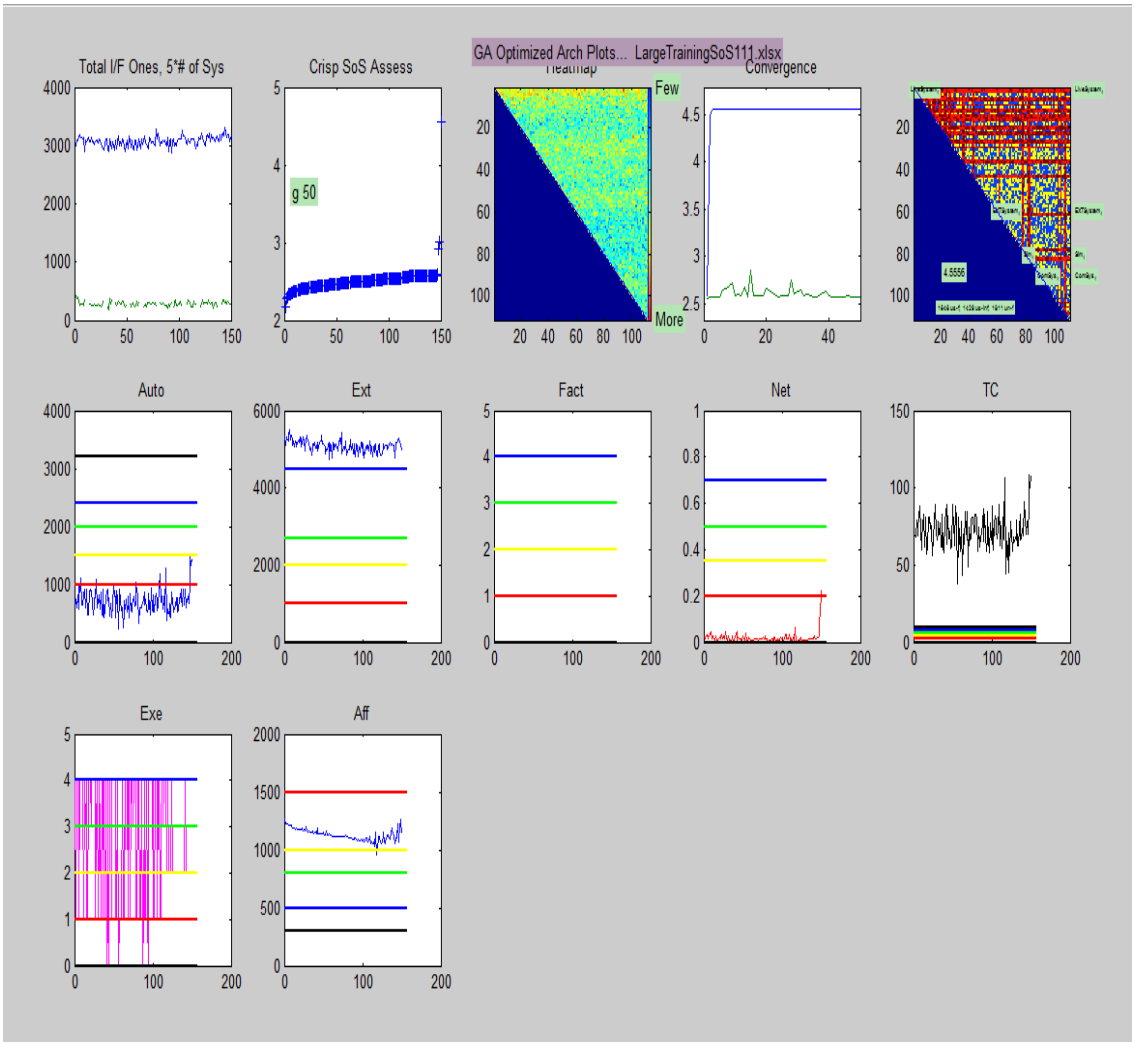


Figure 4.36. Example 111 system example shows bands of less selected systems

5. CONCLUSIONS AND FUTURE WORK

5.1 CONCLUSIONS

A fuzzy genetic method for modeling, assessing, testing and improving SoS architectures was developed using the FILA-SoS meta-architecture. This method generally follows the architecture development method of the DoDAF 2.02 for systems, extended to SoS. Several hypothetical but reasonable examples of various sizes were analyzed by following the method to show its viability through application. It should come as no surprise that modeling an SoS architecture is a lengthy and complex task. The approach of eliciting key attributes through conversations with stakeholders, and building, sharing and vetting models of those attributes that depend on the SoS architecture produces a high payoff in understanding. This understanding extends to the problem domain, potential SoS solutions, and numerous issues facing the SoS designers. Combining the attribute evaluations through a fuzzy inference system, also based on discussions with stakeholders, is a powerful tool to help stakeholders understand trade spaces, impacts of their demands, and opportunities not previously apparent.

Several new techniques were pioneered in this research. The usefulness of the upper triangular form of the meta-architecture, something that seems so obvious now, took a long time to discover. A definition of robustness (for SoS) involving the least loss of functionality for the SoS, after losing participation of any single system, was developed and implemented. This definition of robustness could easily be extended by recursing it to the loss of any number of systems. A generic algorithm for solving the extended FDNA Toy problem was found. All the Matlab code for the implementation of these techniques is listed in Appendices B and C.

The research showed that building models of acknowledged SoS architectures may be accomplished using the generalized method in a real world system example in the proprietary LVC training SoS. The method is helpful in discovering and defining issues, exploring ways to satisfy conflicting stakeholder needs, and in showing the impact of policies (through the rules) on architecture selection and evolution. Key performance attributes that depend on participation in the meta-architecture can be discovered through facilitated interactions with stakeholders and SMEs. The modeling approach can be reused across similar SoS domains with minor modifications. A subset of all, but a still useful group, of KPAs can be defined such that they do depend strongly on the participation in the meta-architecture. Relatively simple fuzzy rule-based systems can combine the KPA evaluations to an overall SoS assessment. The fuzzy genetic approach has been demonstrated to be viable for finding good solutions to several SoS architecting problems under a restrictive meta-model of simple, undirected network graphs representing the system interfaces. This was extended to the directed network in the extended FDNA MITRE Toy problem.

Setting the boundaries of the membership functions, and scaling them independently, is a good way to get rapid understanding about the SoS architecting problem. Because it is tedious to reprogram the Matlab Fuzzy Toolbox with new boundaries, the variable scaling discussed in section 3.3.3 shows how a type of mapping between fuzzy and real world variables can be accomplished quickly and easily in different but related problem domains. This also allows reused solutions which appear similarly shaped in the fuzzy domain but mapped differently in the real domain. By following the map, switching between fuzzy and real values provides a rapid approach

for answering questions about an architecture analysis, or for presenting results to stakeholders in the most understandable way tailored to their specific concerns.

5.2 FUTURE WORK

Extensions of the method in the areas of partial (or perhaps half-hearted) participation by the systems, instead of binary (all or nothing) participation seems to be possible and a fruitful area to investigate. Introducing more uncertainty in the attribute membership functions through use of Type II fuzzy sets or by differently shaped membership functions seems promising for certain types of problems. The process of finding 'good' suggested architectures through application of the fuzzy genetic approach appears to be useful for proposing an SoS architecture. When following the wave model of evolution of an acknowledged SoS, assessing the realizable, negotiated SoS architecture can aid the update plan for the next epoch. Investigations into finding the 'best' shape for membership functions either from the stakeholder discussions or from additional exploration of the trade space seem well warranted.

FILA-SoS research continues by building improved negotiation models and an attractive graphic user interface for building the SoS model. These steps will allow the software to be used in a new SERC sponsored SoS virtual laboratory. The fuzzy assessor approach continues to be used in the latest series of SERC research tasks on an SoS for control of counterfeit parts risk to major DoD weapons systems. The systems in this SoS include, among others: original equipment manufacturers, vendors in supply chains, parts brokers, part retesting standards, the FBI, the Customs service, the military services, and the Justice Department. Making more practitioners aware of the entire FILA-SoS approach, and how to implement the approach on common problems is proposed through

short courses for industry and tutorials that could be provided at several annual systems engineering conferences and workshops. Additionally, the FILA-SoS approach is being used in a graduate-level systems architecting course at Missouri S&T.

APPENDIX A

DETAILED GULF WAR PERFORMANCE MODEL

Performance – for the Gulf War ISR Domain example is made up of surveillance coverage in area per hour and wavelength region, combined with ability to reach the site of a discovered but fleeting high value target before it disappears.

- Background Assumptions: 100,000 square miles in which to hide; 30 minutes from start to finish for an operational launch; on the order of 60 TELs operational; an individual TEL might hide for several days, so the probability of an individual TEL popping out to make a launch is only about 10% per day.

  Rules for combining capabilities into performance:

- Fighters can provide modest capability in non-traditional ISR with on board sensors, *and* deliver several weapons types, but they cost more to operate than many other systems and are relatively poor at ISR tasks
- Remote Piloted Aircraft (RPAs) can provide better ISR capabilities with somewhat less speed and single weapon capabilities, but also require a control station for each 2 RPAs. They are considerably cheaper to operate than fighters
- JSTARS can provide considerable radar ISR capability, and LOS and BLOS relay, but no weapons
- DSP can provide reliable notice of an actual launch over the entire search area, which means there definitely was a TEL in the open at that launch point, but it does not provide very precise localization of the launch point, meaning some search is still required upon an armed vehicle's arrival in the vicinity, and it takes a few minutes to receive the data from DSP. The TEL can hide quickly after launch, leaving not much time to arrive there, find and attack it before it disappears again. In the performance model, the DSP coverage was multiplied by

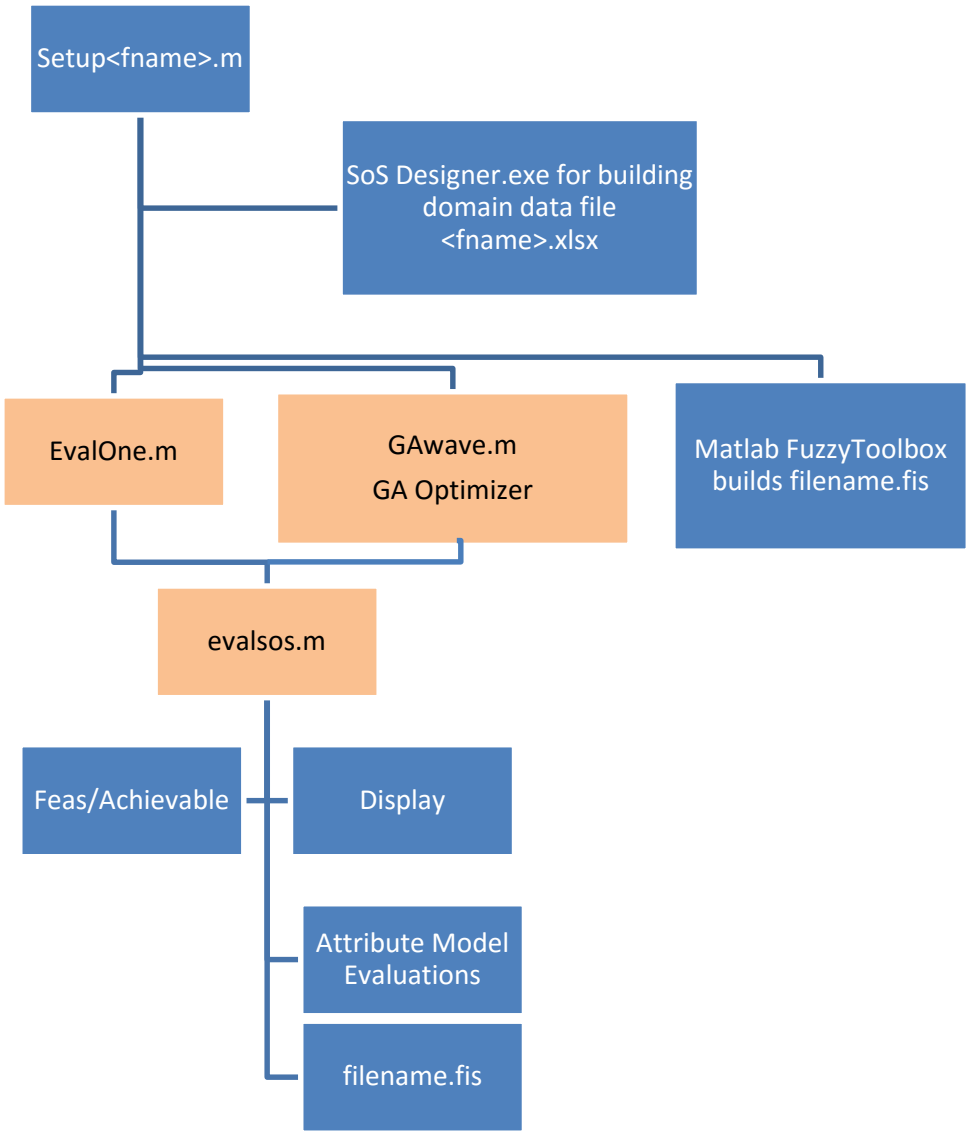
0.01 to account for the likely lack of closure from a DSP detection. DSP is basically free to operate, because it is used for other purposes

- U-2 or Satellite can cover a large area with high resolution, but turnaround time is hours; participation of U-2 or Satellite effectively decreases total area to be searched by other ISR platforms by a reasonable percentage by ruling out certain areas, but does not affect real time surveillance success
- The area to be covered is divided into sectors by the number of participating surveillance systems
- Time to arrive is proportional to the square root of the sector area being covered by each type of system, plus some time for transmitting data to, and double checking by, the 'exploit' systems to insure the target is valid and not in a restricted area
- Probability of successful engagement is defined as 50% if the coverage rate is the total area in half an hour by all the systems, and the time to arrive for an attack after detection is less than 10 minutes. Fighters or RPAs making the discovery are able to attack relatively quickly, transit time is typically less than 5 min for fighters airborne in the adjacent sector, 10 min for RPAs; other types of detection require transit time for the attack vehicle which may be longer if it is in a different sector.

APPENDIX B

MATLAB CODE

Figure B-1. Structure of genetic algorithm and fuzzy assessor Matlab code:



i

Table B-1. Structure of input data and Matlab files

• <fname>.xlsx	Domain input data file. Filled in by hand to start, but now there is a GUI program to fill it in easier with previews of the data to reduce typos Input data parameters may need to be tweaked together with mapfuz membership function edge inputs to get a reasonable model. 5 to 7 specifically named sheets must be included in the input data file
• GAwave.m	Returns a GA 'optimized' good architecture instance (chromosome) in excel file <fname>.xlsx
• evalsos.m	Inputs a chromosome, and probem type; outputs the crisp assessment and evaluation of each individual attribute
• Feas.m	Calculates the achievability matrix using the system number of the first of the common communications systems at the bottom of the systems list
• Attributes	List of items to evaluate an architecture against; combined in rules of fuzzy inference system, and named in the domain input file
• SoSRules	Embedded in Matlab fuzzy inference system files `Fuzzeval44.fis, lvc.fis, sumonly.fis, ToyProb.fis` Very simple…don't pick any worst ones; all good is excellent; performance and affordability trump robustness & flexibility.
• Dispfech.m	Input a chromosome and a achievability matrix; output is a graph of systems & interfaces, and assessment

• mapfuz	Input matrix that maps the range of each attribute gradation to the fuzzy values; currently uses g +1 values for g gradations;
• EvalOne	Reads in the input domain file with a chromosome, outputs the linear chromosome, attribute evaluations, and overall SoS assessment to the same file
• fdn22Atoy.m	Generic FDNA solver called inside evalsos.m
• ReadIn.m	Reads in five Excel sheets of information; system characteristics, capabilities, either or both the upper triangular and the linear form of the chromosome, and a control sheet for the fuzzy inference system and genetic algorithm
• setup<fname>.m	Sets the filename for ReadIn.m, and the number of chromosomes to try in valueExplore.m
• penalty.m	Provides a penalty/reward for the exponent of the netcentric boost in the performance
• valueExplore.m	Biases the chromosomes in a population from few to many randomly placed ones to help explore the SoS design space; plots the data out similar to the GA

INITIAL GA MUTATION PROCESS

Early work on the GA routine was done with a 'try a little of everything' approach. Two mutation processes are imposed on the sorted population of chromosomes. The single best chromosome is always retained, along with less good chromosomes down to about the 20^{th} percentile of the population. The lowest three retained chromosomes are replaced by the chromosome at the 40^{th}, 60^{th}, and 80^{th} percentile. Position of the chromosomes in this adjusted quintile is then randomized.

This group of chromosomes is then replicated four times to fill out the next generation's starting population. A selectable parameter, Delta, typically around 1% or 2%, is the threshold for a uniformly distributed random number generator to decide to mutate each bit of each chromosome in the first quintile of the population. In the second quintile, the decision to mutate a bit is made twice as likely (rnd < (2 times Delta). Sexual crossover is performed at a random position for a random length substring of bits between the third and fourth quintile of chromosomes to generate the population segments for the next generation. In the last quintile of the population, a string of random length, starting at a random position, is transposed with the following bit string within each chromosome.

Any reason for preferring any bit positions or genes within the meta-architecture chromosome, such as the first m bits representing the systems, fell apart in the definition of robustness, where any entire system would be removed as part of the evaluation. The choice of all three methods of mutation was deemed appropriate to insure a broader exploration of the space by the GA. The size of Delta, Population and number of Generations may be selected to complement each other to provide quicker execution or fuller coverage of the space. It is felt that the selection of a linearly biased number of 'one' bits in the initial population speeds up the convergence over a purely random set of initial chromosomes.

FINAL RANKED ROULETTE GA ALGORITHM

Some of the GA literature suggests that a ranked roulette based algorithm, with higher fitness valued chromosomes having a higher likelihood of propagating, may be a faster converging GA approach (Kumar and Jyotishree 2012). The current version of GAwave.m uses the ranked roulette based algorithm. Only the highest fitness

chromosome is guaranteed to be in the next generation. In the initial version of the GA algorithm, a greater proportion of the higher ranked chromosomes in each generation were propagated to the next generation. This change did not seem to change the rate of convergence, but it does seem to lower the average assessment of the remainder of the population using the new algorithm.

The remainder of this appendix is a listing of all the Matlab Code used in the project. Each Matlab function or subroutine starts on a new page.

```
function [chdisp, mov] =dispfech(ch1,fe,crisp);
% version 1 aug 2015    Lou Pape    FILA-SoS
% creates a color coded display of an m system chromosome square,
w/evaluation,
% and a frame of a movie
% given a chromosome, the feasibility matrix, and the evaluation
warning('off');
global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens delta probtype lin
prot;

xywh=zeros(1,4);    % size of display screen array set up
dia=zeros(1,m);     % more array setup...
dia=1:m;
chdisp=zeros(m);    % size the color grid to display
grn=0;              % set counters to zero
rd=0;
bl=0;

for i=1:m           % check feasibility & usage of each interface
    for j=i:m
        if fe(i,j)==0
            if ch1(i,j)==0
                chdisp(i,j)=64;%good - dark brown, unused and
infeasible-good
            else
                chdisp(i,j)=55;%bad -red, used but infeas
                rd=rd+1;
            end
        else
             if ch1(i,j)==0
                chdisp(i,j)=12;%toobad, could have done better - blue
                bl=bl+1;
            else
                chdisp(i,j)=40;%just right, yellow/Green, ideal
                grn=grn+1;
             end
        end
    end
end


xywh=get(gcf,'Position');  %this gets the size of the window, if it's
been changed
set(gcf, 'Position',xywh); %this "sets" the window size for the
getframe below

image(chdisp);              % shows the color codes for each sys &
interface
hold on                     % then type labels & summ. data about this
chrom on it
typ= systyp{1}(1:3);
     widx=xywh(3)-xywh(1);
     hty=xywh(4)-xywh(2);
for i=1:m-1
```

```
  if ~strcmp(systyp{i}(1:3),typ) % if type changes, then print
      text(m+(m/26),i,systyp(i),'BackgroundColor',[.7 .9
.7],'FontSize', (xywh(4)/(2.5*80)) );  %rt lable
      text(  i-(length(char(systyp{i}))/7)-1.5  , i
,systyp{i},'BackgroundColor',[.7 .9 .7],'FontSize', xywh(4)/(2.5*80) );
%lft lbl
      typ=systyp{i}(1:3);
  end
end

plot(dia,dia);             % a reminder line on the systems (diagonal)

text(.2*m,.8*m, num2str( crisp ),'BackgroundColor',[.7 .9
.7],'FontSize', (xywh(4)/120) );
            %now print how many green(used, feas) interface, bad(
used, infeas), &
            %could be better (unused but feasible) interfaces...and
print them
text(.16*m, .95*m, [ num2str(grn) ' us-f;   ' num2str(rd) ' us-inf;   '
num2str(bl) ' un-f']...
       ,'BackgroundColor',[.7 .9 .7],'FontSize', (xywh(4)/170) );
hold off
mov=getframe(gcf);         %save a movie frame for each gen's or
iteration's picture
end
```

```
%Eval and assess One Chromosome, write out chrom, evaluations &
assessment to
%sheet Architecture_Chromosome

global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
    CoD_mat SoD_mat g attr h fisfile pop gens delta probtype lin prot;

mm2=m*(m+1)/2;

[chrom]=Readin(fname);

[mf mff crisp]=evalsos(chrom);

%% write the chromosome & evaluations out to the chromosome sheet
blk=['c7:' num2col(2+mm2) '7'];
xlswrite(fname, chrom, 'Architecture_Chromosome', blk );  %fuzzy
numbers
xlswrite(fname,[mff'] ,'Architecture_Chromosome',['b9:b' num2str(8+g)]
) ;
xlswrite(fname,crisp,'Architecture_Chromosome','b7') ;

attrlabe=cell(2,1);
attrlabe=cellstr(['Arch';'Qual']);
xlswrite(fname,attrlabe,'Architecture_Chromosome','b5:b6') ;
attrlabe=cell(1,1);
attrlabe=cellstr(['Architecture']);
xlswrite(fname,attrlabe,'Architecture_Chromosome','a7') ;
attrlabe=cellstr(['Fuzzy']);
xlswrite(fname,attrlabe,'Architecture_Chromosome','b8') ;
attrlabe=cellstr(['Real']);
xlswrite(fname,attrlabe,'Architecture_Chromosome','c8') ;

attrlabe=cell(g,1);
attrlabe=cellstr(attr);   %attribute names in a column...
xlswrite(fname,attrlabe,'Architecture_Chromosome',['a9:a' num2str(8+g)
]); ;

xlswrite(fname,[mf'],'Architecture_Chromosome',['c9:c' num2str(8+g)] )
;
%           real values, col c

attrlabe=cell(3,1);   %GA control vars
attrlabe=cellstr(['gens';'popu';'delt']);
xlswrite(fname,attrlabe,'Architecture_Chromosome','f9:f11') ;
xlswrite(fname,[gens ; pop ; mu  ],'Architecture_Chromosome','e9:e11')
;

xlswrite(fname,[clock],  'Architecture_Chromosome','h9:m9') ; % put the
date/time of the run on it, too
xlswrite(fname,cellstr(fname), 'Architecture_Chromosome', 'h10');

fclose('all');
```

```
function [mf, mff ,crisp]=evalsos(chro)
%%evaluates each attribute for the chromosome and other input domain
data
warning('off');
global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens delta probtype lin
prot;

fismat=readfis(fisfile{1});
% fdna attributes are:  perfratio, afford (inv of cost), sod,
singleptfail...
% SAR, ISR, attr are perf, afford, flex, robust
% LVC attr are AU   Ext   FaSupt    NetRead TrCap   ExerSupted
Affordability

mm2=m*(m+1)/2;
sc=lin2sc(chro,m);
fe=feas(sc);
%%                   now the case statements for each type of problem
switch probtype

    case 'SAR'
%%
%performance...
cover=0;
maxcover=sum(capsys(:,:)*perf(:,1));

for i=1:m
    for j=1:n
       cover=cover+chro(i)*capsys(j,i)*perf(i,1);
       if perf(i,5)==j    % add a double helping for the main
capability
           cover=cover+chro(i)*perf(i,1)*capsys(j,i);
       end
       end
end
per=cover/maxcover;  %fraction of all ones for systems perfsum
(capsys(:,:)*perf(:,1))
per=per*(1-bump)^penalty(fe,sc);
%%

%affordability
cost=0;

for i=1:m
    cost=cost+perf(i,2)*sc(i,i) +perf(i,3)*(sum(sc(:,i))+sum(sc(i,:))-
2*sc(i,i));
    %     sum of ops cost of system plus interface ccost for each
interface
    %     (minus counting the system twice in sum of i/f row & col
end
%%

%singlept failure in sources of capability test for flexibility
```

```
flx=capsys*chro(1:m)';  % how many systems with each capability
flex=0;
for i=1:n
    if flx(i)<2
       flex=flex+1;
    end
end
%%
%robustness - steps through repetitively to subtract a system and all
it's interfaces, recheck
%performance...

maxloss=0;
loss=zeros(m,1);
for k=1:m
  test=sc;  %start with original
  test(:,k)=0;
  test(k,:)=0;  %sets the kth sys & it's interfaces to zero

  fe=feas(test);
  cover=0;
   for i=1:m
       for j=1:n
       cover=cover+test(i,i)*capsys(j,i)*perf(i,1);
       end
   end
   perr=cover/maxcover;
   perr=perr*(1-bump)^penalty(fe,test);
loss(k)=per-perr;  % per should usually be bigger than perr
end
maxloss=max(loss);

mf=zeros(h,1);
mff=mf;  %also zeroes
mf=[per, -cost,  -flex, -maxloss ];  %real world values, negative if
better closer to zero
for i=1:g    %for each attribute
    mff(i)=map2fuz(mapf(i,:),  mf(i) );
end

mf=[per, -cost, -flex, -maxloss];
for i=1:g    %for each attribute
    mff(i)=map2fuz(mapf(i,:),  mf(i) );
end

crisp=evalfis([mff],fismat);  %ending in f is scaled to 0-h (# of mf's)



%%
    case 'ISR'

        cover=0;
maxcover=sum(capsys(:,:)*perf(:,1));
```

```
for i=1:m
    for j=1:n
       cover=cover+chro(i)*capsys(j,i)*perf(i,1);
    end
end
per=cover/maxcover;  %fraction of all ones for systems perfsum
(capsys(:,:)*perf(:,1))
per=per*(1-bump)^penalty(fe,sc);
%%

%affordability
cost=0;
sc=lin2sc(chro,m);
for i=1:m
    cost=cost+perf(i,2)*sc(i,i) +perf(i,3)*(sum(sc(:,i))+sum(sc(i,:))-
2*sc(i,i));
    %     sum of ops cost of system plus interface ccost for each
interface
    %     (minus counting the system twice in sum of i/f row & col
end
%%

%singlept failure in sources of capability test for flexibility
flx=capsys*chro(1:m)';
flex=0;
for i=1:n
    if flx(i)<2
       flex=flex+1;
    end
end
%%
%robustness - steps through repetitively to subtract a system and all
it's interfaces, recheck
%performance...

maxloss=0;
loss=zeros(m,1);
for k=1:m
  test=sc;  %start with original
  test(:,k)=0;
  test(k,:)=0;  %sets the kth sys & it's interfaces to zero

  fe=feas(test);
  cover=0;
   for i=1:m
       for j=1:n
       cover=cover+test(i,i)*capsys(j,i)*perf(i,1);
       end
   end
   perr=cover/maxcover;
   perr=perr*(1-bump)^penalty(fe,test);
loss(k)=per-perr;
end
maxloss=max(loss);

mf=zeros(h,1);
```

```
mff=mf;
mf=[per, -cost,  -flex, -maxloss ];  %real world values

for i=1:g    %for each attribute
    mff(i)=map2fuz(mapf(i,:),  mf(i) );
end

crisp=evalfis([mff],fismat);  %ending in f is scaled to 0-h (# of mf's)


%%
    case 'LVC'
    switch m    %22 systems requires different evaluation algorithm
than 111
        case 22

            %% calculate au's real value

au=0;
for i=15:20   %only the au systems
    %add together sys present, capabilities of each, plus feasible
    %interespces;  do we add a multiplier, or subtract infeasible
    %interespces??

    for j=1:4    %add first: all the capabilities not controller, not
comm sys
        if sc(i,i)==1     %system is present
           au= au+capsys(j,i) ;
        end
    end     %then add the feasible interfaces to anywhere
    for j=i+1:m   %sum the row of interespces; could do till com1...
        au=au+sc(i,j)*fe(i,j);  %not feasible, don't count
    end
    for j=i-1:-1:1  %sum the column of interfaces going up
        au=au+sc(j,i)*fe(j,i);
    end
end

%%  calculate extens

ex=0;
for i=1:m     %now count everything hooked to hi capacity comms; cap *
fe interface
    for j=i+1:m
        %for k=60:61   %hla=97, dis=98 as systems
        k=4;
           if capsys(k,i)==1 ||  capsys(k,j)==1
             ex=ex+sc(i,j);%*fe(i,j);  removed the feasibility
          % end
        end
    end
end


%% fact support    does not consider feasibility
```

```
fac(1,4)=zeros;

for i=4:5 % large groups
  if sc(i,i)==1
          for j=i+1:com1
            fac(1)=fac(1)+sc(i,j)*fe(i,j);
          end
          for j=i-1:-1:1
            fac(1)=fac(1)+sc(i,j)*fe(i,j);
          end
  end
end
for i=8:9 %med large groups
  if sc(i,i)==1
          for j=i+1:com1
            fac(1)=fac(1)+sc(i,j)*fe(i,j);
          end
          for j=i-1:-1:1
            fac(1)=fac(1)+sc(i,j)*fe(i,j);
          end
  end
end
for i=10:10   % mid level
  if sc(i,i)==1
    for j=i+1:com1
        fac(1)=fac(1)+sc(i,j)*fe(i,j);
    end
    for j=i-1:-1:1
        fac(1)=fac(1)+sc(i,j)*fe(i,j);
    end
  end
end    % fac(1) now has feasible interfaces with large & med

for i=1:2
  if sc(i,i)==1
    for j=i+1:com1
        fac(2)=fac(2)+sc(i,j)*fe(i,j);
    end
    for j=i-1:-1:1
        fac(2)=fac(2)+sc(i,j)*fe(i,j);
    end
  end
end    % fac(2) now has feasible interfaces with outside groups

for i=6:7
   if sc(i,i)==1
      for j=i+1:com1
        fac(3)=fac(3)+sc(i,j)*fe(i,j);
      end
      for j=i-1:-1:1
        fac(3)=fac(3)+sc(i,j)*fe(i,j);
      end
   end
end    % fac(3) has feasible interfaces with different outside groups
```

```
for i=10:com1-1
     if sc(i,i)==1
      for j=i+1:com1
        fac(4)=fac(4)+sc(i,j)*fe(i,j);
      end
      for j=i-1:-1:1
        fac(4)=fac(4)+sc(i,j)*fe(i,j);
      end
     end
end

%fac(4) has all the remaininig interfaces
fa=0;
if fac(1)>3   %traditional
    fa=fa+1;
    if fac(2)> 3    %new combined;  can't get to higher one unless you
have enough below
       fa=fa+1;
       if fac(3) >5   % outside groups
           fa=fa+1;
           if fac(4) >30   %everyone
               fa=fa+1;
           end
       end
    end
end

%fa now has 4 if everyone has sufficient interfaces


%% netreadiness

nr=0;
nr=sum(sum(sc .* fe));   %% sum of used feasible interfaces
nr=2*nr*(1+bump)^penalty(sc, fe); %give it a netcentric bump for
feasible interfaces (again?)
nr=nr/mm2;

%%  training capabilities

tc=0;
for i=1:com1
    tc=tc+sc(i,i)*perf(i,1);
end   %sum of present systems times their relative value in the input
domain data


%% exercises supported

es=0;
esp=zeros(1,4);
%if sc(1,1)==1
for i=15:19
    esp(1)=esp(1)+sc(i,i);   %counts au's, too;
end
```

```
for i=4:5
    esp(2)=esp(2)+sc(i,i);  % large
end
for i=1:2:3
    esp(3)=esp(3) +sc(i,i);   % hq
end
for i=2:2
    esp(4)=esp(4)+sc(i,i);   %counts facts present (not talking to them
tho)
end
for i=6:7
    esp(4)=esp(4)+sc(i,i);   %counts facts present (not talking to them
tho)
end
esp(4)=esp(4)+sc(2,2);  %adds other groups

if esp(1)>0  %     can't get to next level unless enough of lower
    es=es+1;
    if esp(2)> 0    % high level
       es=es+1;
       if esp(3) >0    % mid level
           es=es+1;
           if esp(4) >0   % outsiders
               es=es+1;
           end
       end
    end
end     % es now must have some of each below to get one above
%end   %can only get exercises supported if exercise monitor is there!
%% affordability
af=0;
for i=1:m
    af=af+sc(i,i)*perf(i,3);       %counts operating cost for systems
    for j=i+1:m
       af=af+sc(i,j)*perf(i,2);    %counts interfacing cost per
interface
    end
end

%summarizing:
[mf]=[au, ex, fa, nr, tc, es, -af];
for i=1:g    %for each attribute
    mff(i)=map2fuz(mapf(i,:),  mf(i) );
end
crisp=evalfis([mff],fismat);  %ending in f is scaled to 0-h (# of mf's)

        case 111

            %% calculate au's real value

au=0;
for i=60:90   %only the au systems
    %add together sys present, capabilities of each, plus feasible
    %interespces;  do we add a multiplier, or subtract infeasible
    %interespces??
```

```
    for j=2:53    %add first: all the capabilities not controller, not
comm sys
        if sc(i,i)==1     %sys is present
           au= au+capsys(j,i) ;
        end
    end     %then add the feasible interfaces to anywhere
    for j=i+1:m   %sum the row of interespces; could do till com1...
        au=au+sc(i,j)*fe(i,j);  %not feasible, don't count
    end
    for j=i-1:-1:1  %sum the column of interfaces going up
        au=au+sc(j,i)*fe(j,i);
    end
end

%%  calculate extens

ex=0;
for i=1:m     %now count everything hooked to hi capacity comms; cap *
fe interface
    for j=i+1:m
       for k=60:61   % hi cap comm sys
           if capsys(k,i)==1 ||  capsys(k,j)==1
             ex=ex+sc(i,j);%*fe(i,j);  removed the feasibility
           end
        end
    end
end


%% fact support    does not consider feasibility

fac(1,4)=zeros;

for i=21:25 % large groups
  if sc(i,i)==1
          for j=i+1:com1
            fac(1)=fac(1)+sc(i,j)*fe(i,j);
          end
          for j=i-1:-1:1
            fac(1)=fac(1)+sc(i,j)*fe(i,j);
          end
  end
end
for i=28:32 %med large groups
  if sc(i,i)==1
          for j=i+1:com1
            fac(1)=fac(1)+sc(i,j)*fe(i,j);
          end
          for j=i-1:-1:1
            fac(1)=fac(1)+sc(i,j)*fe(i,j);
          end
  end
end

for i=2:9
```

```
  if sc(i,i)==1
    for j=i+1:com1
        fac(2)=fac(2)+sc(i,j)*fe(i,j);
    end
    for j=i-1:-1:1
        fac(2)=fac(2)+sc(i,j)*fe(i,j);
    end
  end
end    % fac(2) now has feasible interfaces with all if's

for i=10:20   % mid level
  if sc(i,i)==1
    for j=i+1:com1
        fac(3)=fac(3)+sc(i,j)*fe(i,j);
    end
    for j=i-1:-1:1
        fac(3)=fac(3)+sc(i,j)*fe(i,j);
    end
  end
end    % fac(3) now has feasible interfaces with large & med

for i=26:27
     if sc(i,i)==1
      for j=i+1:com1
        fac(4)=fac(4)+sc(i,j)*fe(i,j);
      end
      for j=i-1:-1:1
        fac(4)=fac(4)+sc(i,j)*fe(i,j);
      end
     end
end

for i=33:com1-1
     if sc(i,i)==1
      for j=i+1:com1
        fac(4)=fac(4)+sc(i,j)*fe(i,j);
      end
      for j=i-1:-1:1
        fac(4)=fac(4)+sc(i,j)*fe(i,j);
      end
     end
end %fac(4) has all the remaininig interfaces

%fac(4) has all the remaininig interfaces
fa=0;
if fac(1)>0   %traditional
    fa=fa+1;
    if fac(2)> 0    %new combined;  can't get to higher one unless you
have enough below
       fa=fa+1;
       if fac(3) >0   % outside groups
           fa=fa+1;
           if fac(4) >0   %everyone
               fa=fa+1;
           end
       end
```

```
    end
end

%fa now has 4 if everyone has sufficient interfaces


%% netreadiness

nr=0;
nr=sum(sum(sc .* fe));   %% sum of used feasible interespces
nr=2*nr*(1+bump)^penalty(sc, fe); %give it a netcentric bump for
feasible interfaces (again?)
nr=nr/mm2;

%%  train capabilities

tc=0;
for i=1:com1
    tc=tc+sc(i,i)*perf(i,1);
end   %sum of present systems times their relative value in the input
domain data


%% ex supported

es=0;
esp=zeros(1,4);
%if sc(1,1)==1
for i=15:19
    esp(1)=esp(1)+sc(i,i);   %counts au's, too;
end
for i=4:5
    esp(2)=esp(2)+sc(i,i);  % large
end
for i=1:2:3
    esp(3)=esp(3) +sc(i,i);   % hq
end
for i=2:2
    esp(4)=esp(4)+sc(i,i);   %counts facts present (not talking to them
tho)
end
for i=6:7
    esp(4)=esp(4)+sc(i,i);   %counts facts present (not talking to them
tho)
end
esp(4)=esp(4)+sc(2,2);  %adds other groups

if esp(1)>0  %     can't get to next level unless enough of lower
    es=es+1;
    if esp(2)> 0    % high level
       es=es+1;
       if esp(3) >0    % mid level
           es=es+1;
           if esp(4) >0   % outsiders
               es=es+1;
```

```
          end
       end
    end
end     % es now must have some of each below to get one above
%end   %can only get exercises supported if exercise monitor is there!
%% affordability
af=0;
for i=1:m
    af=af+sc(i,i)*perf(i,3);       %counts operating cost for systems
    for j=i+1:m
       af=af+sc(i,j)*perf(i,2);    %counts interfacing cost per
interface
    end
end

%summarizing:
[mf]=[au, ex, fa, nr, tc, es, -af];
for i=1:g    %for each attribute
    mff(i)=map2fuz(mapf(i,:),  mf(i) );
end
crisp=evalfis([mff],fismat);  %ending in f is scaled to 0-h (# of mf's)

    end

    case 'FDN'

        % fdna attributes are:  perfratio, afford (inv of cost), sod,
singleptfail...

%perfrat:
[cod,sod,p25]=fdn22Atoy( CoD_mat, SoD_mat , 25, chro);
[cod,sod,p100]=fdn22Atoy( CoD_mat, SoD_mat , 100, chro);
prat=p25(m)/p100(m);

%affordability
cost=0;

for i=1:m
    cost=cost+perf(i,2)*sc(i,i) +perf(i,3)*(sum(sc(:,i))+sum(sc(i,:))-
2*sc(i,i));
end

%sod
sd=sod(m);  %it was calculated in fdn22toy, above

%singlept failure
sing=capsys*chro(1:m)';
spf=0;
for i=2:n-1
    if sing(i)<2
       spf=spf+1;
    end
end
mf=zeros(h,1);
mff=mf;
```

```
mf=[prat, -cost, -spf, -sd ];  %real world values
for i=1:g    %for each attribute
    mff(i)=map2fuz(mapf(i,:),  mf(i) );
end

crisp=evalfis([mff],fismat);  %ending in f is scaled to 0-h (# of mf's)

end

%%
end
```

```
function [cod,sod,P]=fdn22Atoy(CoD_mat,SoD_mat,P1,chro)
%% chro is the fila-sos linear chromosome
%SoD_matx is input as strength of dependency, 0<element<1, and controls
for
%loops in the sod matrix, not the chromosome, which can be random
%where for sys i, interface (i,j)i<j, i is a feeder node to j, and
feeding goes
%clockwise, and j depends on i [i<j is the upper triangular portion]
%if j depended on i, then the dependency would appear in the lower
portion
%of the full matrix in SOD and COD
%    if no dependency, then perf of the node is in the input domain or
%    characteristics sheet under the system performance
%m is the number of systems
%
warning('off');
global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens delta probtype lin
prot;

arch=lin2sc(chro,m);      %an upper tri matrix, random interfaces,
input matrix to evaluate
full=arch+triu(arch,1)';  %make into adjacency matrix by filling out
the bottom
COD=CoD_mat.*full;    %cod & sod matrix design protect from loops
SOD=SoD_mat.*full;    %not having a 1 in the interfaces deletes the
cod/sod
%%
%I'm a receiver node if I have any entries in my vertical interfaces
% column; figure out how much I receive with sod & cod...
P=perf(:,1);     % from system characteristics in input domain data
P(1)=P1;    % vary start node performance as input
for k=1:4             % because I use original Perf values, and they're
recalculated during FDNA
                      % at line 63-74
feedn=zeros(m,1); %column matrix
sod=zeros(m,1);   %column matrix
sodp=zeros(m,1);   %column matrix
cod=zeros(m);     % square

%piter=zeros(m,11);   % used only for troubleshooting during
development
%piter(:,1)=P(:);
%%
for j=1:m                                  %col 1 of sod should be zeroes
    for i=1:j-1
        if SOD(i,j)~=0 && chro(i)~=0  && chro(j)~=0  %then sys i feeds
j
            feedn(j)=feedn(j)+1;          % how many feed j that are
less than j?
                                          % found an i that feeds j, add
one
                                          % to count of items that feed
j
```

```
            sodp(j)=sodp(j)+SOD(i,j)*P(i);%100*(1-SoD_mat(i,j));  % sum
will be divided by number in feedn
            sod(j)=sod(j)+SOD(i,j);
            cod(i,j)=P(i)+COD(i,j);  % cod is a new efficiency based on
criticality
                                        % it will be used later to
find the
                                        % minimum, for the new P(i)
        end
    end
    for i=j+1:m
        if SOD(i,j)~=0   && chro(i)~=0  && chro(j)~=0  % then j feeds i
            feedn(j)=feedn(j)+1;        % how many feed i that are
greater than i
            sodp(j)=sodp(i)+SOD(i,j)*P(i);%+100*(1-SoD_mat(i,j));  %
sum will be divided by number in feedn
            sod(j)=sod(j)+SOD(i,j);
            cod(i,j)=P(i)+COD(i,j);  % cod is a new efficiency based on
criticality
                                        % it will be used later to
find the
                                        % minimum, for the new P(i)
        end

    end

    c2d=cod(:,j);        % whole list of feeders with criticality to
receiver j
                         % for taking minimum of to continue

    if feedn(j)>0
        if min(size(c2d(c2d~=0)))>0
           P(j)=min((sodp(j)/feedn(j))+(1-sod(j)/feedn(j))*100,  min(
c2d(c2d~=0) )   ) ;

        else
           P(j)=(sodp(j)/feedn(j))+((1-sod(j)/feedn(j))*100);

        end
    else
        P(j)=min(P(j), 100);

    end
    if feedn(j)~=0
        sod(j)=sod(j)/feedn(j);     % think this is necessary
    end
end
%piter(:,k+1)=P(:);     % used during development
end
%% sum(piter)             % used during development
end
```

```
function fe=feas(sch);
%%
% modified for fdna problems 30 Jul 2015
% this works right; checked on small files 10Jul13
% fe will be the feasibility matrix, generated from common
communication
%      systems interfaces among the other systems, or for fdna if sod
%      exists
% m is the number of systems in the chromosome
% sch is the square chromosome matrix itself
% com is first comm system number; comm systems are in the right hand
% columns
warning('off');
global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens delta probtype lin
prot;

fe=zeros(m);                % also, if none work, zeroes going back
if probtype=='FDN'     % then they can feed and receive, somewhere

  for i=1:m
     for j=1:m
        if (sch(i,i)==1 && sch(j,j)==0) && (CoD_mat(i,j)~=0 &&
SoD_mat(i,j)~=0)
            fe(i,j)=1;   %sod, cod  equal 0, then no connection
        end
     end
  end

else     % feas depends on comm unit interfaces with other systems
         % and not on CoD
  for i=1:m;
    fe(i,i)=sch(i,i);  %systems are copied over; if they exist, they're
feasible
  end

  for i=com1:m;
    if sch(i,i)==1 ;    %comm system i is present then feas is
possible, else not
        for j=1:com1-1;
           for k=(j+1):com1-1 ;
             if (sch(j,j)==1) && (sch(k,k)==1) ; %both systems are
present
               fe(j,k) =fe(j,k) || sch(j,i)*sch(k,i);% 'or' the other
comm links
                        %   both sys also i/f to comm a||b||etc, then
fe=1
             end
           end
        end
    end
end

for j=com1:m;          % finish up with within the comm systems
   for k=1:j-1;
```

```
       if (sch(j,j)==1) && (sch(k,k)==1)  %both system and comm sys are
present
          fe(k,j) =fe(k,j) || 1;% 'or' the other comm links both 1,
then fe=1
       end
   end
end

end

end
```

```
%% a genetic algorithm routine to find the best chromosome
%fname is the excel file for the "good" chromosome, and attribute
evaluations
%    and crisp assessment of the SoS
% fname is also the domain data file with all the SoS system data
%
% Offer_Status.xlsx is the negotiated member systems (first wave,
all=0)
% file, in the first column, but the participating systems is NOW also
in the
% first column of the Characteristics sheet for running the optimizer
or
% the single chromosome assessor (GAwave.m or evalsos.m)
%   The following variables control the architecture:
% m is number of systems
% n is number of component capabilities
% probtype three letter code, and linearinput for a chromosome  in the
linear
%  form (1) or upper triangular matrix form (0)
% system names, types (if used), major capability (if used), interface
develop cost,
%   operations cost, performance in the major capability, and
development
%   time are all in the Characteristics sheet of fname.xlsx in named
%   columns
% - system vs. capability matrix and number of capabilities, are on the
%   Capabilities sheet
% - when not in linear form, the input chromosome is on sheet
Interfaces
%   (the chromosome is always output to sheet Architecture_Chromosome
% - mutation rate, delta is the probability of mutating each bit - (1%
to 5%
%   seems about right) but also used for deciding how long and where to
%   transpose; bump is the interoperability/netcentric boost, amount of
%   penalty increase for infeasible/unachievable interface, penup &
%   decrease (reward) for achievable/feasible interface, pendn; are all
on
%   sheet FuzzyGA, with the .FIS filename, the number of attributes,
and
%   the number of membership functions.  com1 is the system number of
the
%   first communication system (should always be grouped at the end of
the
%   list)
% - p is the number of chromosomes in a population for one generation
% - gens is the number of generations to run
% - mapf is the matrix of attributes and fuzzy membership function
%  crossing points - fuzzy values are 0 (bad) to number of MFs (best)
%    now includes interfaces from negotiations, too, by reading
offerstatus
%   and keeping any interfaces associated with kept systems
% - reads in the offer status file for next waves; all zeroes for first
wave

% Lou Pape,  2015oct5
%
tym=now;
```

```
warning('off'); % it interferes with making it an executable
global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens delta probtype lin
prot;

[initch]=Readin(fname); % reads in all the system/capability data, and
neg chromosome if it exists
                        % in the Characteristics sheet; checks to see
if
                        % Offer_stat exists in addition to fname.xlsx
                        % protected systems

scrsz = get(0,'ScreenSize');  %set up plot figure size fairly large,
but to fit the screen
figure('Position',[60 scrsz(4)/25 scrsz(3)/1.2 scrsz(4)/1.18]);
set(gcf, 'Position',[60 scrsz(4)/25 scrsz(3)/1.2 scrsz(4)/1.18]); %full
window on double screen w/taskbar on left

mm2=m*(m+1)/2; %number of total bits in chromosome
numm=size(mapf);
nummfs=numm(2)-1;
attv=zeros(g,1);
attvf=attv;
chrom=zeros(pop,mm2);
assess=zeros(pop,1);
stat=zeros(gens,2);  % for plotting convergence at end

% plotting constants
heat=zeros(1,mm2);   % to store base of heatmap
heattot=heat;        % for final,overall heatmap
frac=.7;       % how deep to reach for plotting the heatmap from the
best
for ki=1:m
     sip(ki)=11+ (44*ki/m); % total color range, from min to max
end
dia=zeros(1,m);    % for plotting a line through the 'system' squares
dia=1:m;
col=['k' 'r' 'y' 'g' 'b' 'k' 'r' 'y' 'g' 'b' 'k' 'r' 'y' 'g' 'b'];
    r=2;             % sets up rows & columns of display screens
    c=5;             % based on number of attributes in fuzzy evals
  if g>5
    r=3;
  end
pltsym=['-' '--' 'r' 'k' 'm' 'b'];

%setup variables for later plotting
plo=zeros(pop,g+7);     % iii or pop

% this part handles the negotiated baseline from last wave - can't let
that
%  mutate and evolve!
% add this in for waves...and interfaces - init ch is the starting
%  negotiated chromosome read in from linear or interface UT form.
%   prot is the systems negotiated from last wave
```

```
% create protection chromosome with any neg. system, and any interface
from
% input(negotiated, if > wave 0) that was present for a neg system

keep=zeros(m);
if sum(prot)>0
  initsc=lin2sc(initch,m);  % output from Readin function is linear
form,
                            % this switches back to UTM
  for i=1:m
    if prot(i)==1
        keep(i,i)=1;
    end
    for j=i+1:m
        if prot(i)==1 || prot(j)==1  %if either system is negotiated in
            if initsc(i,j)==1        %then if its interface is a one,
keep it
                keep(i,j)=1;         % should make this a switch to
include
                                     % protecting interfaces OR not...
            end
        end
    end
  end
end
  initch=sc2lin(keep,m);      % initch now has what must be kept from
mutating

clearvars mov;   %sets up to make a  movie of the generations if you
want to watch later
                 %  "implay(mov)" will let you step through it;
"save(filename"
                 %  saves the movie and everything else in a .mat file,
if you like it
%% This runs poprandom m system chromosomes through the fuzzy evaluator
% and picks the best using roulette selection for sexual crossover
% to replay the movie, use    implay(mov)    in the command window
% it runs from within matlab  (ie, not executable) and includes the
% plotting each generation

ch1=zeros(m);        % single square chromosome matrix - plotting


%% initialize a random population to start the GA   chrom (pop, mm2)
for q=1:pop
    for i=1:mm2
 chrom(q,i)=round( q/pop*rand);  %
    end
     % seed the comm systems a little extra:
       for i=com1:m
           if (rand>.5) && (chrom(q,i)==0)  ; %give another .5 chance
to be a one
               chrom(q,i)=1;
           end
       end %seed extra comms
```

```
    %prevent selected (systems from negotiations) from being mutated
    %away now...for initial wave, you must make offer_stat.xlsx all
zeros (no
    %systems selected from negotiations yet (or the first col of
    %Characteristics sheet)
  %
      %***** this  includes the negotiated interfaces
      for x=1:mm2
          if initch(1,x)==1   % hold on to negotiated interfaces
              chrom(q,x)=1;   % if other new, proposed by randoms or
mutations, good!
          end                 %if ever want to consider tiny percentage
      end                     %systems or interfaces will quit, do it
in this loop!
      %end negotiated sys/interfaces  ... every member of the
population has the right ones
  end  % of q stepping through initial random ('cept for wave
holdovers) population

%  we now have a generally random population for generation 1 with
varying numbers of ones
%    AND we've protected any previous wave negotiated ones from being
removed.

%% generational loop - you already have the random starting population
from above
%     including negotiation results, which will be protected through
%     mutations later...
%%
for gen=1:gens   % big outer loop for generations


%   1) EVALUATE  whole pop;   2) SORT whole pop;  4) PLOT sorted pop
statistics;
%      3) rank pop by cum fitness;  5) crossover selected parents to
make new pop of chroms;
%       6) RE-LOOP to step 1 for next generation
%      Start the sorting and plotting process of a population within
each generation
%      just above we randomly initialized the chromosome population
%%
for q=1:pop;   %   prepare to eval, sort then plot as we step through
each member of the population
    plo(q,1)=q;       %plo(1) is the plotting index
    plo(q,2)=sum(chrom(q,m+1:mm2));   %2 is the total number of
interfaes in a chromosome
    plo(q,3)=5*sum(chrom(q,1:m)); %sum of participating systems

   ch1=lin2sc( chrom(q,:),m ); %chrom is linear, ch1 is upper
triangular
   fe=feas(ch1);
    %here's where you call the fuzzy evaluator for each member of the
    %population

   [attv , attvf, crisp]=evalsos(chrom(q,:));  % one at a time
    plo(q,4)=crisp;
```

```
    plo(q,7+1:7+g)=attv(:);


%%for plotting... not if you will be executing this file in the ABM
 %[chdisp, mov(q)]=dispfech(m,ch1,fe,crisp);   future function...

    %crisp is the fuzzy evaluation


end
% whole population is now evaluated and stored for sorting

%% sort section
heat=zeros(1,mm2);
chrom=[chrom plo(:,4)]; %adds the fitness column to end of chrom in pop

chrom=sortrows(chrom, -(mm2+1) );  % sorts the chromosome population on
that column in descending order
fitnorm=sum(chrom(:,mm2+1));  %adds column of sos fitnesses to do the
normalization
chrom(:,mm2+1)=chrom(:,mm2+1)/fitnorm;  %normalized fitnesses, highest
fitness at top

for ii=2:pop
    chrom(ii,mm2+1)=chrom(ii,mm2+1)+chrom(ii-1,mm2+1);  %now the
fitness column is the cumulative, normalized fitness
end
plo=sortrows(plo,-4);  %sorts all the plotted values in descending
order of fitness, too

plo(:,1)=1:pop;  % renumbers the index column for plotting all rows
(pop)in order of fitness, not place in the generation

%%
%    now display the population plots for this generation

%% plotting section
set(gcf, 'Position',[60   scrsz(4)/25 scrsz(3)/1.2 scrsz(4)/1.18]);
%full window on double screen w/taskbar on left

subplot(r,c,1);
plot(plo(:,1), plo(:,2),'-',plo(:,1), plo(:,3),'--'), title('Total I/F
Ones, 5*# of Sys'); %number of total ones in chrom, systems+i/f's
%
subplot(r,c,2);
   plo=sortrows(plo,4);  %sorts all the plotted values in ascending
order of fitness, too
   plo(:,1)=1:pop;  % renumbers the index column for plotting all rows
(pop)in order of fitness, not place in the generation
plot(plo(:,1), plo(:,4),'+'), title('Crisp SoS Assess'); %crisp output
of evaluator/assessor
   plo=sortrows(plo,-4);  %sorts all the plotted values in descending
order of fitness, too
```

```
   plo(:,1)=1:pop;  % renumbers the index column for plotting all rows
(pop)in order of fitness, not place in the generation

hold on
text(pop/20,.8*max(plo(:,4)),['g ' num2str(gen)],'BackgroundColor',[.7
.9 .7] );  % put gen no near top left corner
hold off


%
subplot(r,c,3);       % 3 heatmap, 4,5convergence, & best from each
generation
  %heat
    heat=sum(chrom(1:round(pop*frac),1:mm2)); % add ones as deep as
frac
    heattot=heattot+heat;
    if gen<gens
      hot=max(heat);
      cold=min(heat);
    else      % we are on the last generation...
      hot=max(heattot);
      cold=min(heattot);
      heat=heattot;
    end
   ext=hot-cold;   %blue is 12, red is 55, range = 43
   heat=12. + ((heat-cold)/ext)*43. ; %scale min/max whole array to
appropriate color

   image( [lin2sc(heat,m) sip' sip'] );  % plots the values in the
upper triang form, with scale on rt side
hold on
x=zeros(1,m);
x(1,:)= m+.5;
plot(x, dia);    %plots a line at the right edge of the heatmap
plot(dia,dia),title('Heatmap');
text(m+2.3+(m/26),m,'More','BackgroundColor',[.7 .9 .7] );  % hot label
text(m+2.3+(m/26),1,'Few','BackgroundColor',[.7 .9 .7] );  %cold label
 text(-m/8,-m/7,['GA Optimized Arch Plots...  '
fname],'BackgroundColor',[.7 .6 .7] );

hold off
%end of heatmap

% converg if appl would be best of each gen in plot slot 4
stat(gen,1)=plo(1,4);   % best crisp of this gen
stat(gen,2)=plo(round(pop/10),4);  % one tenth of the way down from
best (top 10%)

if gen==gens
subplot(r,c,4);
mg=min(min(.9*stat(:,1)) );
xg=max(max(stat)*1.05 );
rg=xg-mg;
plot(1:gens,stat), axis([0 gens mg xg]), title('Convergence');
end
```

```
%% rest of the attributes, best chrom to worst.. .

   plo=sortrows(plo,4);  %sorts all the plotted values in ascending
order of sos fitness, for plotting
   plo(:,1)=1:pop;  % renumbers the index column for plotting all rows
(pop)in order of fitness, not place in the generation

for j=1:g
subplot(r,c,j+5);
plot(plo(:,1), abs(plo(:,7+j)) ,pltsym(j) ), title(attr(j));   % all
attributes in this loop
hold on
  for i=1:h+1           % now show mf boundary lines
    plot([1 round(1.05*pop)], [abs(mapf(j,i))
abs(mapf(j,i))],'color',col(i),'LineWidth',2)
  end
hold off
end

   plo=sortrows(plo,-4);  %sorts all the plotted values in descending
order of fitness, for selection in tournament
   plo(:,1)=1:pop;  % renumbers the index column for plotting all rows
(pop)in order of fitness, not place in the generation


subplot(r,c,5);
%plot color display of chromosome at end of each generation
ch1=lin2sc(chrom(1,1:mm2),m); %best chrom in this gen (not cum, norm
fitness column)
fe=feas(ch1);
crisp=plo(1,4);   %already know this one - best of the lot, top one
[chdisp, mov(gen)]=dispfech(ch1,fe,crisp);  %makes a movie, too


%% for next generation, get ready by creating next gen population from
old roulette winners

if gen<gens
popu=zeros(pop+1,mm2); %size the new population array one bigger than
P
popu(1,:)=chrom(1,1:mm2);  %save the best one (not including rank col)

for i=2:2:pop
    cho=rand;
     pt1=find(chrom(:,mm2+1)>=cho,1);  %chrom still sorted by rank
column at end
     p1=chrom( pt1,1:mm2);  %picks the first one with cum fitness >=
rand()
    cho=rand;
     p2=chrom( find(chrom(:,mm2+1)>=cho,1),1:mm2);  %finds another one
>=cho
    cho=rand;
     xo=max(1,round((mm2-1)*cho));  %at a random point in the
chromosome.. .
```

```
     cho=rand;
    if cho>.9
        p1=1-p1;  %invert each BIT in one parent - on rare occasions,
        % since number of bits is how I plot things; this would allow
wider
        % variations in number of bits, whereas transposition alone
does not
    end
    popu(i,:) = [p1(1:xo) p2(xo+1:mm2)]; % cross over the parents parts
    popu(i+1,:) = [ p1(xo+1:mm2) p2(1:xo)]; % to make 2 new offspring
end

%allow a chance to randomly mutate all but best one again
for i=2:pop;
    for j=1:mm2;
        if rand > 1-delta;
            popu(i,j)=1- popu(i,j);   %inverts a single bit
         end
    end
end
% oh - must set mutated positions back to negotiated ones, again - now,
after
% mutations, if they changed...
  for i=1:pop
    for x=1:mm2    % counting both systems - and interfaces...
        if initch(1,x)==1  %first wave, initch is all zeroes, never
happens
            popu(i,x)=1;
        end
    end
  end;

  %%allowed them to mutate in popu generation, now returned them to
proper belonging

  chrom(:,1:mm2)=popu(1:pop,:);  %all pop rows of chrom and all chrom
bits of pop for new generation
  chrom=chrom(:,1:mm2);   % insures it deletes the pesky sos assess
column at the end of chromosome for next gen

end  % of if not enough generations yet

end % of gens
%%




%%   format and write output files
%% write the chromosome & evaluations out to the chromosome sheet
 [attv , attvf, crisp]=evalsos(chrom(1,1:mm2)); %evaluates final best
one again to write out the values
blk=['c7:' num2col(2+mm2) '7'];
xlswrite(fname, chrom(1,1:mm2), 'Architecture_Chromosome', blk );
%fuzzy numbers
```

```
xlswrite(fname,[attvf'] ,'Architecture_Chromosome',['b9:b'
num2str(8+g)] ) ;
xlswrite(fname,crisp,'Architecture_Chromosome','b7') ;

attrlabe=cell(2,1);
attrlabe=cellstr(['Arch';'Qual']);
xlswrite(fname,attrlabe,'Architecture_Chromosome','b5:b6') ;
attrlabe=cell(1,1);
attrlabe=cellstr(['Architecture']);
xlswrite(fname,attrlabe,'Architecture_Chromosome','a7') ;
attrlabe=cellstr(['Fuzzy']);
xlswrite(fname,attrlabe,'Architecture_Chromosome','b8') ;
attrlabe=cellstr(['Real']);
xlswrite(fname,attrlabe,'Architecture_Chromosome','c8') ;

attrlabe=cell(g,1);
attrlabe=cellstr(attr);   %attribute names in a column...
xlswrite(fname,attrlabe,'Architecture_Chromosome',['a9:a' num2str(8+g)
]); ;

xlswrite(fname,[attv'],'Architecture_Chromosome',['c9:c' num2str(8+g)]
) ;
%           real values, col c

attrlabe=cell(4,1);   %GA control vars
attrlabe=cellstr(['gens';'pop ';'delt']);
xlswrite(fname,attrlabe,'Architecture_Chromosome','f9:f11') ;
xlswrite(fname,[gens ; pop ; delta
],'Architecture_Chromosome','e9:e11') ;

xlswrite(fname,[clock],  'Architecture_Chromosome','h9:m9') ; % put the
date/time of the run on it, too
xlswrite(fname,cellstr(fname), 'Architecture_Chromosome', 'h10');

fclose('all');


tym=86400*(now-tym);
disp(['it took ' num2str(tym) ' seconds'])
```

```
%%  if you want to spend the time (30-40 seconds) to label the
chromosome output file
% fname must already exist, and the linear chromosome must be the
rightmost
% sheet in Excel file 'fname' (all my attempts at using the sheet name
result
% in "index exceeds matrix dimensions".  it uses m, from input file,
too.

global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens delta probtype lin
prot;


    %%   format and write output lin chrom labels
mm2=m*(m+1)/2; %number of total bits in chromosome
labe=cell(1,mm2);   %*******creates the label values for the output
chromosome Excel file
k=0;
for i=1:m;
    k=k+1;
    labe{1,k}=['S' num2str(i)];   % systems
end
 for i=1:m;
     for j=i+1:m
    k=k+1;
    labe{1,k}=['i' num2str(i) '-' num2str(j)];   % interfaces
     end
 end   %**********************end of creating the label array for excel
sheet

%labels cells just above the chromosome with labe matrix just created
above
blk=['c6:' num2col(5+mm2) '6'];
xlswrite(fname, labe, 'Architecture_Chromosome', blk);  % that's the
big array of excel.cell labels

fclose('all');  %don't have it already open for activex

%% now label colors on top of labels from above
%  Connect to Excel   all the active x in one place because it's not
% compatible with xlswrite simultaneously
Excel = actxserver('excel.application');
% Get Workbook object
WB = Excel.Workbooks.Open(fullfile(pwd, fname),0,false);
WB.Worksheets.Item(WB.Sheets.Count).Activate

cel=2;  %arch qual - text is below
WB.Worksheets.Item(WB.Sheets.count).Range([num2col(cel) '5:'
num2col(cel) '6']).Interior.ColorIndex = 6;%bright yellow?
WB.Worksheets.Item(WB.Sheets.count).Range([num2col(9) ':' num2col(mm2)
]).Columns.ColumnWidth = 3 ;

cel=3;   %do the systems background color...
```

```
WB.Worksheets.Item(WB.Sheets.count).Range([num2col(cel) '5:'
num2col(cel+m-1) '6']).Interior.ColorIndex = 7;%purple?
cel=m+2;
col=4;
for i=1:(m-1)
    cels=cel+1;
    for j=(i+1):m
        cel=cel+1;
    end %now cels is the first/start cell for the interface, cel is the
last one
WB.Worksheets.Item(WB.Sheets.count).Range([num2col(cels) '5:'
num2col(cel) '6']).Interior.ColorIndex = col;% start green?
    if col==4
        col=8;
    else
        col=4;
    end %alternates color at the end of each range of interfaces
end %of overall nested loop on color alternates
% Save Workbook
WB.Save();
% Close Workbook
WB.Close();
% Quit Excel
Excel.Quit();

%% now the text labeling...
 cel=3;   %systems...
sysint=cell(1,1);
sysint=cellstr(['Systems']);
xlswrite(fname,sysint,'Architecture_Chromosome',[ num2col(cel) '5']) ;

cel=m+2;
for i=1:(m-1)
    cels=cel+1;
    for j=(i+1):m
        cel=cel+1;
    end %now cels is the first/start cell for the interface, cel is the
last one
sysint=cellstr(['Interfaces to Sys ' num2str(i) ]);
xlswrite(fname,sysint,'Architecture_Chromosome',[ num2col(cels) '5']) ;

end     %of overall nested loop on color alternates
fclose('all');
%end
```

```
function sc = lin2sc(llin,m)
% takes the linear chromosome of m systems and m(m-1)/2 interfaces
% returns square matrix sc size m upper triangular with systems
% on diagonal
sc=zeros(m);
for i=1:m
    sc(i,i)=llin(i);
end
k=m;
for i=1:m
    for j=(i+1):m;
        k=k+1;      %counter for position in the chromosome
        sc(i,j)=llin(k);
    end;
end

end   %of the function
```

```
function y=map2fuz(mfs, inpu)  % lowerlimit, then upperlimits of mfs
%% version jul2015 for Serc Toy prob
% maps to the fuzzy variable from the real variable input as inpu
% mfs is the array of nummfs+1 MF upper bounds (except first element is
lower bound)
%  assume MF fuzzy values starts at zero, (count-1)=num of mfs
% negative values if better is a lower abs value (nearer zero), such as
% cost for affordability; lower cost is better
quit=0;
yy=0;
numm=size(mfs);
nummfs=numm(2)-1;  %how wide the mfs array is (starts at 0, so one more
than)
mf=mfs;
inp=inpu;

 if (inp) >=  (mf(nummfs+1) )% beyond high end
    yy=nummfs;
    quit=1;
 end
 if (inp) <=  (mf(1) )% beyond low end
    yy=0;
    quit=1;
 end

for i=1:nummfs   %mf(1)=lower limit, mf(5)=high end of mf(when
nummfs=4)
 if quit==0   %haven't exceeded a mf crossover point yet
    if inp<(mf(i+1))   %inpu is less than the next larger crossover...
       yy=(inp-(mf(i)))/((mf(i+1)-mf(i)))+(i-1);  %fuzzy vars are size
1
       quit=1; %after you find one, you're done; don't need to change
anymore
    end
 end
end

y=yy; %return the value calculated in the fuzzy domain
end
```

```
function [blkstr] = num2col(x)
%calculates the column letterlabel in excel from the column number
% 5 Jul 14 10pm version
%Lou Pape, RT-109
%up to  475255    26^4+26^3+26^2+26+1
% it's math with no zero placeholder, like roman numerals
b=26; % should you ever wish to change the base of the calculation
b0=b^0;  % could just use one, thereby saving one run time
comcputation,
        % but this keeps the pattern
b1=b^1;   % 26
b2=b^2;   % 676
b3=b^3;   % 17576
b4=b^4;   % 456976
b10=b1+b0;   %26 + 1
b20=b2+b10;   %676 + 26 + 1
b30=b3+b20;
%b40=b4+b30;
[blkstr]=char.empty(1,0);
%this would be a nice place to do fancy error handling, but I simply
return
% an answer that is not nonsense
x=fix(x);
if x<1
    [blkstr] =['A'];
else
    if x>(b30*b)    % too big; excel handles only 3 letters deep
       [ blkstr]=[ 'WRONG'];
    else   %now check the valid range

if x>b20*b   % greater than AAAA-1, or ZZZ; and, too big for excel A -
XFD
    xx=x-b20;
    xy=fix(xx/b3); % how many b^3 in thereafter removing zzz
    [blkstr]=[blkstr char(64+xy)];  % blkstr starts out empty, this is
leftest letter
    x=x-xy*(b3);   %what's remaining after leftmost digit
end
if x>b10*b   %greater than AAA-1=zz; remaining after b3s are counted
        xx=x-b10;
        xy=fix(xx/b2);
        [blkstr]=[blkstr char(64+xy)];  %adds next "digit" of column
name
        x=x-xy*(b2);
end
 if x>b
     xx=x-1;
     xy=fix(xx/b);
     [blkstr]=[blkstr char(64+xy)];
     x=x-xy*(b);
 end
     [blkstr]=[blkstr char(64+x)];
end
end
end
```

```
function pen=penalty(fe,ch)

global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens delta probtype lin
prot;

 pen=0;
 for i=1:m;
       for j=i+1:m;  % try only interfaces, not systems
            if fe(i,j) > ch(i,j);   %it's feasible, you didn't use it
               pen=pen+pendn*.5;       % blue color   mminor penalty -
bad
            else
                if fe(i,j)==ch(i,j);
                    pen=pen-pendn;  %you used feasibility rightly green
- less penalty - good
                else
                    pen=pen+penup; %it's infeasible but you used it,
the worst: red more penalty
                end
           end
       end
 end
end    % big penalty, bad if 1-bump raised to it; more perf, better;
```

```
function [chrom]=Readin(fname)
%%
%reads in the domain data from the new GUI format
%give it the filename as a parameter...that's all
% reads following sheets of input domain data gui output:
%  Characteristics, Capabilities, Initerfaces, Architecture_Chromosome,
SOD, COD,
%  FuzzyGA
%reads all in at once, closes the file, then sorts it out to globals...
warning('off');
[chnum chtxt]=xlsread(fname, 'Characteristics');

[canum catxt]=xlsread(fname, 'Capabilities');

if chnum(2,4)==1 % input is in linear format
    [ionum iotxt]=xlsread(fname, 'Architecture_Chromosome');
else
    [ifnum iftxt]=xlsread(fname, 'Interfaces');
end

if chtxt{2,5}=='FDN'
  [conum cotxt]=xlsread(fname, 'COD');
  [sonum sotxt]=xlsread(fname, 'SOD');
end

[fnum ftxt]=xlsread(fname, 'FuzzyGA');
fclose('all');

%%

global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens delta probtype lin
prot;

m=chnum(1,1);  %from Characteristics sheet
nchar=chnum(2,1);
n=canum(2,1);
probtype=chtxt{2,5};
lin=chnum(2,4);

systyp=chtxt(7:6+m,1);
prot=chnum(6:5+m,1);      % if protected is from updated inputs (must
be, if adding new systems)
     neg=zeros(m,1);
     [neg]=xlsread('Offer_Stat.xlsx');
     if sum(neg)>0        % if negotiations protect some systems, they
will be non zero in offerstat
                          % default offerstat has all zeros, so if
                          % negotiations doesn't change it, then any
protected
                          % systems are from input domain data
         prot=neg;
     end
perf=chnum(6:5+m,3:2+nchar);    % includes costs, too, now
```

```
capsys=canum(6:5+m,1:n)';  %from Capabilities sheet
capname=catxt(6,2:1+n);

if probtype=='FDN'
    CoD_mat=conum(6:5+m,1:m);
    SoD_mat=sonum(6:5+m,1:m);
end

if lin==1
    chrom=ionum(6:6, 2:1+(m*(m+1)/2 ));  %the input chromosome, linear
format
else
    chsc=ifnum(6:5+m,1:m);  %input chromosome is in UTmatrix form, in
Interfaces sheet
    chrom=sc2lin(chsc,m);   % force it into the linear form, but not in
the excel sheet, yet
end

fisfile=ftxt(2,2);
g=fnum(2,1);                % g is how many attributes
attr=ftxt(7:6+g,1);         % attribute names
h=fnum(3,1);                % # of MFs in each attribute
mfname=ftxt(6,3:2+h);
mapf=fnum(6:5+g, 1:1+h);
com1=fnum(4,1);
pop=fnum(1,7);
gens=fnum(1,8);
delta=fnum(2,7);
bump=fnum(2,8);
penup=fnum(3,7);
pendn=fnum(3,8);


end
```

```
function lin = sc2lin(sc,m);
% takes the square matrix sc size m upper triangular with systems on
diagonal
% returns linear chromosome lin of m systems and m(m-1)/2 interfaces
% (total  1 by m(m+1)/2 )
lin=zeros(1,m*(m+1)/2);
for i=1:m ;
    lin(1,i)=sc(i,i);
end  ;
k=m;
for i=1:m ;
    for j=(i+1):m;
        k=k+1;      %counter for position in the chromosome
        lin(1,k)=sc(i,j);
    end;
end;

end   %of the function
```

```
%setup fila-sos  l pape   30 Jul 2015
%%

fname='Toy24Jul15.xlsx'
global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens mu probtype lin;

%[chrom]=Readfdna(fname); % reads in ALL the background data from GUI
data input


iii=160;   %default number of random chromosomes for value explore
mm2=(m+1)*m/2;
```

```
%setup fila-sos  l pape   30 Jul 2015
%%
fname='Toy24Jul15one.xlsx'

global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens mu probtype lin;


iii=300;   %default number of random chromosomes for value explore
mm2=(m+1)*m/2;
```

```
%setup fila-sos  l pape   30 Jul 2015
%%
fname='Toy24Jul15one2.xlsx'

global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens mu probtype lin;


iii=300;   %default number of random chromosomes for value explore
mm2=(m+1)*m/2;
```

```
%setup fila-sos  l pape   30 Jul 2015
%%

fname='isr.xlsx'
global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens mu probtype lin;

%[chrom]=Readin(fname); % reads in ALL the background data from GUI
data input


iii=200;   %default number of random chromosomes for value explore
mm2=(m+1)*m/2;
```

```
%setup fila-sos  l pape   30 Jul 2015
%%

fname='LargeTrainingSoS.xlsx'
global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens mu probtype lin;

%[chrom]=Readin(fname); % reads in ALL the background data from GUI
data input


iii=100;   %default number of random chromosomes for value explore
mm2=(m+1)*m/2;
```

```
%setup fila-sos  l pape   30 Jul 2015
%%

fname='LargeTrainingSoS111.xlsx'
global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens mu probtype lin;

%[chrom]=Readin(fname); % reads in ALL the background data from GUI
data input


iii=200;   %default number of random chromosomes for value explore
mm2=(m+1)*m/2;
```

```
%setup fila-sos  l pape   30 Jul 2015
%%

fname='LargeTrainingSoS22.xlsx'
global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens mu probtype lin;

%[chrom]=Readin(fname); % reads in ALL the background data from GUI
data input


iii=100;   %default number of random chromosomes for value explore
mm2=(m+1)*m/2;
```

```
%setup fila-sos  l pape   30 Jul 2015
%%

fname='SAR29.xlsx'
global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens mu probtype lin;

%[chrom]=Readin(fname); % reads in ALL the background data from GUI
data input


iii=150;   %default number of random chromosomes for value explore
```

```
%setup fila-sos  l pape   30 Jul 2015
%%
fname='SAR29one.xlsx'
global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens mu probtype lin;


iii=150;   %default number of random chromosomes for value explore
```

```
%setup fila-sos  l pape   30 Jul 2015
%%

fname='SAR29sum.xlsx'
global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens mu probtype lin;

%[chrom]=Readin(fname); % reads in ALL the background data from GUI
data input


iii=250;   %default number of random chromosomes for value explore
```

```
%not a function valueExplore(iii,fname);
%% This runs    iii   random system chromosomes through the fuzzy
evaluator
   %  (set iii and fname before running this with the 'setupxxxx.m')
% and plots them so you can set the values for the edges of the
membership function
% it takes about tenth of a second for each chromosome
% run this, then check distribution of the membership function
boundaries on the
% distribution of values for each attribute in the command window.
Make
% adjustments as desired either in the Excel file or the GUI, save, and
% repeat
% Version: all exe's  2015Aug20
% Lou Pape, RT-109

warning('off');

tym=now;
global m n sys systyp capname capsys perf bump mapf com1  penup pendn
...
      CoD_mat SoD_mat g attr h fisfile pop gens delta probtype lin
prot;

[chromdummy]=Readin(fname);        %linear string form

%%  setting up constants
scrsz = get(0,'ScreenSize');  %set up  figure size fairly large
figure('Position',[60 scrsz(4)/25 scrsz(3)/1.2 scrsz(4)/1.18]);
mm2=m*(m+1)/2; %number of total bits in chromosome
numm=size(mapf);
nummfs=numm(2)-1;
attv=zeros(g,1);
attvf=attv;
chrom=zeros(iii,mm2);
plo=zeros(iii,g+7);     % iii or pop
clf
heat=zeros(1,mm2);
heatsc=heat;
frac=.4;       % how deep to reach for plotting the heatmap
for ki=1:m
     sip(ki)=11+ (44*ki/m); % color range, from min to max
end
dia=zeros(1,m);    % for plotting a line through the 'system' squares
dia=1:m;
col=['k' 'r' 'y' 'g' 'b' 'k' 'r' 'y' 'g' 'b' 'k' 'r' 'y' 'g' 'b'];
    r=2;             %set up rows & columns of display screens
    c=5;
  if g>5
    r=3;
  end
pltsym=['-' '--' 'r' 'k' 'm' 'b'];

%%
for q=1:iii                   % create & evaluate random strings
(architectures)one at a time
```

```
    chrom(q,:)=round( .9*q/iii+randn(1,mm2)/3);
       % seed the comm systems to have  more 1s even for low numbered
chroms:
       for i=com1:m
           if rand>.5
               chrom(q,i)=1;
           end
       end

    for x=1:mm2    %this is necessary for the normal generated
chromosome
        if chrom(q,x)<0
            chrom(q,x)=0;
        end
        if chrom(q,x)>1
            chrom(q,x)=1;
        end
    end   %CHROMOSOME q of iii generated
    % for visualizeing the chromosome distribution later
      ch1=zeros(m,m);
    plo(q,1)=q;                      % serial number within population
    plo(q,2)=sum(chrom(q,m+1:mm2));   % number of interfaces
    plo(q,3)=5*sum(chrom(q,1:m));     % 5*sum of participating systems

    % here's where you call the fuzzy evaluator
[attv , attvf, crisp]=evalsos(chrom(q,:));  % one at a time

     plo(q,4)=crisp;
     plo(q,7+1:7+g)=attv(:);
  ch1=lin2sc(chrom(q,:),m);
  fe=feas(ch1);
    %plo(q,5&6&7)=placeholder for heatmap(need slider) & convergence &
best(q or gen,crisp)
  heat=heat+chrom(q,:);   % add all the chromosomes to 'heat'
      % save best chrom found so far
      if q==1
          bestchrom=chrom(q,:);
          bestcrisp=crisp;
          bestq=1;
      else
          if crisp>bestcrisp
              bestchrom=chrom(q,:);
              bestcrisp=crisp;
              bestq=q;
          end
      end
end
%% sort section  used for optimization
%s=sortrows(plo,4);   %sort by crisp value
%plo=s;
%plo(:,1)=sort(s(:,1));


%% plotting section
set(gcf, 'Position',[60   scrsz(4)/25 scrsz(3)/1.2 scrsz(4)/1.18]);
%full window on double screen w/taskbar on left
```

```
subplot(r,c,1);
plot(plo(:,1), plo(:,2),'-',plo(:,1), plo(:,3),'--'), title('Total I/F
Ones, 5*# of Sys'); %number of total ones in chrom, systems+i/f's
%
subplot(r,c,2);
plot(plo(:,1), plo(:,4),'+'), title('Crisp SoS Assess'); %crisp output
of evaluator/assessor
%
subplot(r,c,3);       % 3,4,5 heatmap, convergence, &final
  %heat

   hot=max(heat);
   cold=min(heat);
   ext=hot-cold;   %blue is 12, red is 55, range = 43
   heat=12. + ((heat-cold)/ext)*43. ; %scale min/max to appropriate
color

   image( [lin2sc(heat,m), sip' ,sip'] );  % plots the values in the
upper triang form, with scale on rt side
hold on
x=zeros(1,m);
x(1,:)= m+.5;
plot(x, dia);    %plots a line at the right edge of the heatmap
plot(dia,dia),title('Heatmap');
text(m+2.3+(m/26),m,'More','BackgroundColor',[.7 .9 .7] );  % hot label
text(m+2.3+(m/26),1,'Few','BackgroundColor',[.7 .9 .7] );  %cold label
 text(-m/8,-m/7,['EXPLORiNG for MF EDGEs vs. Arch   '
fname],'BackgroundColor',[.7 .9 .7] );
hold off
%end of heatmap

% converg if appl would be best of each gen in plot slot 4
% plot best chrom here:
subplot(r,c,5);
image( 12+25*lin2sc( bestchrom,m));
  hold on
  plot(dia,dia),title('Best');
  text(m+.3+(m/26),m/2, sprintf('# %3g',bestq),'BackgroundColor',[.7 .9
.7] );  %which one label
  text(m+.3+(m/26),3, sprintf('Eval
%.3g',bestcrisp),'BackgroundColor',[.7 .9 .7] );  %assessment label
hold off

for j=1:g
subplot(r,c,j+5);
plot(plo(:,1), abs(plo(:,7+j)) ,pltsym(j) ), title(attr(j));   % all
attributes in this loop
hold on
  for i=1:h+1           % mf lines
    plot([1 round(1.05*iii)], [abs(mapf(j,i))
abs(mapf(j,i))],':','color',col(i),'LineWidth',2)
  end
hold off
end
```

```
%end
hold off
%% Correlation
cor=zeros(iii,4+g);
cor(:,1:4)=plo(:,1:4);
cor(:,5:4+g)=plo(:,8:7+g);
disp( ['   q       ' 'i/f       ' '5*sys      ' 'crisp    ' ]);
disp( [ attr(1:g)]' );
disp(num2str(corrcoef(cor)));
%% current values for membership function edges vs distribution
bou=plo(:,4); %crisp
bou=sort(bou);
disp('crisp');
for i=0:nummfs
    disp(sprintf('MF edge = %.4g but %.4g is the distribution',  i,
bou(round(max(1,i*iii/nummfs)))));
end

for gg=1:g
    bou=plo(:,7+gg);
    bou=sort(bou);
    disp(' ');
    disp( attr(gg));
    for i=0:nummfs
     disp(sprintf('MF edge%.2g = %.4g but %.4g is the distribution', i,
mapf(gg,i+1), bou(round(max(1,i*iii/nummfs)))));
    end

 end
```

APPENDIX C

MATLAB FUZZY INFERENCE SYSTEM (.FIS) FILES

Filename: sumonly.fis

```
[System]
Name='sumonly'
Type='mamdani'
Version=2.0
NumInputs=4
NumOutputs=1
NumRules=4
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='Performance'
Range=[1 4]
NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.18 1.07 0.18 1.36]
MF2='Marginal':'gauss2mf',[0.218 1.82 0.218 2.23]
MF3='Acceptable':'gauss2mf',[0.21 2.75 0.21 3.25]
MF4='Exceeds':'gauss2mf',[0.135 3.64 0.135 3.99]

[Input2]
Name='Affordability'
Range=[1 4]
NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.18 1.07 0.18 1.36]
MF2='Marginal':'gauss2mf',[0.218 1.82 0.218 2.23]
MF3='Acceptable':'gauss2mf',[0.21 2.75 0.21 3.25]
MF4='Exceeds':'gauss2mf',[0.135 3.64 0.135 3.99]

[Input3]
Name='Development-Flexibility'
Range=[1 4]
NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.18 1.07 0.18 1.36]
MF2='Marginal':'gauss2mf',[0.218 1.82 0.218 2.23]
```

MF3='Acceptable':'gauss2mf',[0.21 2.75 0.21 3.25]
MF4='Exceeds':'gauss2mf',[0.135 3.64 0.135 3.99]

[Input4]
Name='Robustness'
Range=[1 4]
NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.18 1.07 0.18 1.36]
MF2='Marginal':'gauss2mf',[0.218 1.82 0.218 2.23]
MF3='Acceptable':'gauss2mf',[0.21 2.75 0.21 3.25]
MF4='Exceeds':'gauss2mf',[0.135 3.64 0.135 3.99]

[Output1]
Name='SoS-Arch-Fitness'
Range=[1 4]
NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.18 1.07 0.18 1.36]
MF2='Marginal':'gauss2mf',[0.218 1.82 0.218 2.23]
MF3='Acceptable':'gauss2mf',[0.21 2.75 0.21 3.25]
MF4='Exceeds':'gauss2mf',[0.135 3.64 0.135 3.99]

[Rules]
1 1 1 1, 1 (1) : 2
2 2 2 2, 2 (1) : 2
3 3 3 3, 3 (1) : 1
4 4 4 4, 4 (1) : 1

Filename: Fuzzeval44.fis

```
[System]
Name='Fuzzeval44'
Type='mamdani'
Version=2.0
NumInputs=4
NumOutputs=1
NumRules=10
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='Performance'
Range=[1 4]
NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.18 1.07 0.18 1.36346302134404]
MF2='Marginal':'gauss2mf',[0.218 1.82 0.218 2.23141985318686]
MF3='Acceptable':'gauss2mf',[0.21 2.75 0.21 3.2487756413035]
MF4='Exceeds':'gauss2mf',[0.135 3.64704559113483 0.135 3.99]

[Input2]
Name='Affordability'
Range=[1 4]
NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.18 1.07 0.18 1.36346302134404]
MF2='Marginal':'gauss2mf',[0.218 1.82 0.218 2.23141985318686]
MF3='Acceptable':'gauss2mf',[0.21 2.75 0.21 3.2487756413035]
MF4='Exceeds':'gauss2mf',[0.135 3.64704559113483 0.135 3.99]

[Input3]
Name='Development-Flexibility'
Range=[1 4]
NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.18 1.07 0.18 1.36346302134404]
MF2='Marginal':'gauss2mf',[0.218 1.82 0.218 2.23141985318686]
```

MF3='Acceptable':'gauss2mf',[0.21 2.75 0.21 3.2487756413035]
MF4='Exceeds':'gauss2mf',[0.135 3.64704559113483 0.135 3.99]

[Input4]
Name='Robustness'
Range=[1 4]
NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.18 1.07 0.18 1.36346302134404]
MF2='Marginal':'gauss2mf',[0.218 1.82 0.218 2.23141985318686]
MF3='Acceptable':'gauss2mf',[0.21 2.75 0.21 3.2487756413035]
MF4='Exceeds':'gauss2mf',[0.135 3.64704559113483 0.135 3.99]

[Output1]
Name='SoS-Arch-Fitness'
Range=[1 4]
NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.18 1.07 0.18 1.36346302134404]
MF2='Marginal':'gauss2mf',[0.218 1.82 0.218 2.23141985318686]
MF3='Acceptable':'gauss2mf',[0.21 2.75 0.21 3.2487756413035]
MF4='Exceeds':'gauss2mf',[0.135 3.64704559113483 0.135 3.99]

[Rules]
1 1 1 1, 1 (1) : 2
4 4 4 4, 4 (1) : 1
2 2 2 2, 1 (1) : 1
3 3 3 3, 4 (1) : 1
4 4 2 -1, 4 (1) : 1
2 2 2 3, 2 (1) : 1
2 2 3 2, 2 (1) : 1
2 3 2 2, 2 (1) : 1
3 2 2 2, 2 (1) : 1
4 4 -1 2, 4 (1) : 1

Filename: ToyProb.fis

```
[System]
Name='ToyProb'
Type='mamdani'
Version=2.0
NumInputs=4
NumOutputs=1
NumRules=10
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='Performance'
Range=[0 4]
NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.24 0.0333 0.20 0.80]
MF2='Mediocre':'gauss2mf',[0.11 1.15 0.126 1.85]
MF3='AboveAvg':'gauss2mf',[0.196 2.27 0.178 2.84]
MF4='VeryGood':'gauss2mf',[0.18 3.30 0.18 3.99]

[Input2]
Name='Affordability'
Range=[0 4]
NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.24 0.0333 0.20 0.80]
MF2='Mediocre':'gauss2mf',[0.11 1.15 0.126 1.85]
MF3='AboveAvg':'gauss2mf',[0.196 2.27 0.178 2.84]
MF4='VeryGood':'gauss2mf',[0.18 3.30 0.18 3.99]

[Input3]
Name='SinglePtFailure'
Range=[0 4]
NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.24 0.0333 0.20 0.80]
MF2='Mediocre':'gauss2mf',[0.11 1.15 0.126 1.85]
MF3='AboveAvg':'gauss2mf',[0.196 2.27 0.178 2.84]
MF4='VeryGood':'gauss2mf',[0.18 3.30 0.18 3.99]

[Input4]
Name='StrengthOfDependency'
Range=[0 4]
```

NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.24 0.0333 0.20 0.80]
MF2='Mediocre':'gauss2mf',[0.11 1.15 0.126 1.85]
MF3='AboveAvg':'gauss2mf',[0.196 2.27 0.178 2.84]
MF4='VeryGood':'gauss2mf',[0.18 3.30 0.18 3.99]

[Output1]
Name='SoS-Arch-Fitness'
Range=[0 4]
NumMFs=4
MF1='Unacceptable':'gauss2mf',[0.24 0.0333 0.20 0.80]
MF2='Mediocre':'gauss2mf',[0.11 1.15 0.126 1.85]
MF3='AboveAvg':'gauss2mf',[0.196 2.27 0.178 2.84]
MF4='VeryGood':'gauss2mf',[0.18 3.30 0.18 3.99]

[Rules]
1 1 1 1, 1 (1) : 2
4 4 4 4, 4 (1) : 1
2 2 2 2, 1 (1) : 1
3 3 3 3, 4 (1) : 1
4 4 2 -1, 4 (1) : 1
2 2 2 3, 2 (1) : 1
2 2 3 2, 2 (1) : 1
2 3 2 2, 2 (1) : 1
3 2 2 2, 2 (1) : 1
4 4 -1 2, 4 (1) : 1

Filename: lvc.fis

```
[System]
Name='lvc'
Type='mamdani'
Version=2.0
NumInputs=7
NumOutputs=1
NumRules=18
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='AU'
Range=[-0.1 5.1]
NumMFs=5
MF1='None':'trimf',[-0.1 0.02381 1.262]
MF2='Minimal':'trimf',[0.03619 1.262 2.5]
MF3='Sufficient':'trimf',[1.262 2.5 3.738]
MF4='Complex':'trimf',[2.5 3.738 4.976]
MF5='Fully':'trimf',[3.738 4.976 5.1]

[Input2]
Name='Ext'
Range=[-0.1 5.1]
NumMFs=5
MF1='Not':'trimf',[-0.1 0.02381 1.262]
MF2='Slight':'trimf',[0.02381 1.262 2.5]
MF3='Sufficient':'trimf',[1.262 2.5 3.738]
MF4='Mostly':'trimf',[2.5 3.738 4.976]
MF5='Fully':'trimf',[3.738 4.976 5.1]

[Input3]
Name='FactSupt'
Range=[-0.1 5.1]
NumMFs=5
MF1='None':'trimf',[-0.1 0.02381 1.262]
```

MF2='Trad':'trimf',[0.02381 1.262 2.5]
MF3='Multi':'trimf',[1.262 2.5 3.738]
MF4='Civil':'trimf',[2.5 3.738 4.976]
MF5='Complete':'trimf',[3.738 4.976 5.1]

[Input4]
Name='Net'
Range=[-0.1 4.1]
NumMFs=5
MF1='VeryInsuff':'trimf',[-0.1 0 1]
MF2='Insufficient':'trimf',[0 1 2]
MF3='Sufficient':'trimf',[1 2 3]
MF4='Good':'trimf',[2 3 4]
MF5='Brilliant':'trimf',[3 4 4.1]

[Input5]
Name='TC'
Range=[-0.1 4.1]
NumMFs=5
MF1='0':'trimf',[-0.1 0 1]
MF2='25':'trimf',[0 1 2]
MF3='50':'trimf',[1 2 3]
MF4='75':'trimf',[2 3 4]
MF5='100':'trimf',[3 4 4.1]

[Input6]
Name='ExSupt'
Range=[-0.1 4.1]
NumMFs=5
MF1='NoSupt':'trimf',[-0.1 0 1]
MF2='Medium':'trimf',[0 1 2]
MF3='Large':'trimf',[1 2 3]
MF4='Larger':'trimf',[2 3 4]
MF5='Largest':'trimf',[3 4 4.1]

[Input7]
Name='Aff'
Range=[-0.1 4.1]
NumMFs=5
MF1='TooExpensive':'trimf',[-0.1 0 1]

MF2='HighCost':'trimf',[0 1 2]
MF3='Marginal':'trimf',[1 2 3]
MF4='Good':'trimf',[2 3 4]
MF5='Excellent':'trimf',[3 4 4.1]

[Output1]
Name='TrainVal'
Range=[0 5]
NumMFs=5
MF1='Bad':'trimf',[-0.2 0 1.27645502645503]
MF2='Poor':'trimf',[0.486507936507936 1.47650793650794 2.44047619047619]
MF3='Good':'trimf',[2.66962962962963 3.59962962962963 4.56962962962963]
MF4='Superb':'trimf',[3.61772486772487 5 5.15]
MF5='Avg':'trimf',[1.54 2.5462962962963 3.53]

[Rules]
1 1 1 1 1 1 1, 1 (1) : 2
2 2 0 0 0 0 0, 1 (1) : 1
0 2 2 0 0 0 0, 1 (1) : 1
0 0 2 2 0 0 0, 1 (1) : 1
0 0 0 2 2 0 0, 1 (1) : 1
3 3 3 -1 0 -1 0, 2 (1) : 1
0 3 3 3 0 0 0, 2 (1) : 1
0 0 3 3 3 0 0, 2 (1) : 1
4 4 4 4 0 0 0, 3 (1) : 1
4 4 4 0 4 0 0, 3 (1) : 1
4 4 4 4 4 4 0, 3 (1) : 1
4 4 4 4 4 4 3, 4 (1) : 1
0 4 4 4 4 0 3, 3 (1) : 1
5 5 5 5 5 5 2, 3 (1) : 1
5 5 5 5 5 5 4, 4 (1) : 1
4 4 4 4 4 4 3, 4 (1) : 1
3 3 3 -1 0 -1 -1, 3 (1) : 1
4 3 5 3 4 3 3, 4 (1) : 2

APPENDIX D

DODAF 2.0 MODEL VIEWPOINT EXPLANATIONS

Models	Descriptions
AV-1: Overview and Summary Information	Describes a Project's Visions, Goals, Objectives, Plans, Activities, Events, Conditions, Measures, Effects (Outcomes), and produced objects.
AV-2: Integrated Dictionary	An architectural data repository with definitions of all terms used throughout the architectural data and presentations.
CV-1: Vision	The overall vision for transformational endeavors, which provides a strategic context for the capabilities described and a high-level scope.
CV-2: Capability Taxonomy	A hierarchy of capabilities which specifies all the capabilities that are referenced throughout one or more Architectural Descriptions.
CV-3: Capability Phasing	The planned achievement of capability at different points in time or during specific periods of time. The CV-3 shows the capability phasing in terms of the activities, conditions, desired effects, rules complied with, resource consumption and production, and measures, without regard to the performer and location solutions.
CV-4: Capability Dependencies	The dependencies between planned capabilities and the definition of logical groupings of capabilities.
CV-5: Capability to Organizational Development Mapping	The fulfillment of capability requirements shows the planned capability deployment and interconnection for a particular capability phase. The CV-5 shows the planned solution for the phase in terms of performers and locations and their associated concepts.
CV-6: Capability to Operational Activities Mapping	A mapping between the capabilities required and the operational activities that those capabilities support.
CV-7: Capability to Services Mapping	A mapping between the capabilities and the services that these capabilities enable.
DIV-1:Conceptual Data Model	The required high level data concepts and their relationships.
DIV-2: Logical Data Model	The documentation of the data requirements and structural business process (activity) rules. In DoDAF V1.5, this was the OV-7.
DIV-3: Physical Data Model	The physical implementation format of the Logical Data Model entities, e.g., message formats, file structures, physical schema. In DoDAF V1.5, this was the SV-11.
OV-1: High Level Operational Concept Graphic	The high-level graphical/textual description of the operational concept.
OV-2: Operational Resource Flow Description	A description of the resource flows exchanged between operational activities.
OV-3: Operational Resource Flow Matrix	A description of the resources exchanged and the relevant attributes of the exchanges.
OV-4: Organizational Relationships Chart	The organizational context, role or other relationships among organizations.

Models	Descriptions
OV-5a: Operational Activity Decomposition Tree	The capabilities and activities (operational activities) organized in an hierarchal structure.
OV-5b: Operational Activity Model	The context of capabilities and activities (operational activities) and their relationships among activities, inputs, and outputs; Additional data can show cost, performers or other pertinent information.
OV-6a: Operational Rules Model	One of three models used to describe activity (operational activity). It identifies business rules that constrain operations.
OV-6b: State Transition Description	One of three models used to describe operational activity (activity). It identifies business process (activity) responses to events (usually, very short activities).
OV-6c: Event-Trace Description	One of three models used to describe operational activity (activity). It traces actions in a scenario or sequence of events.
PV-1: Project Portfolio Relationships	Describes the dependency relationships between the organizations and projects and the organizational structures needed to manage a portfolio of projects.
PV-2: Project Timelines	A timeline perspective on programs or projects, with the key milestones and interdependencies.
PV-3: Project to Capability Mapping	A mapping of programs and projects to capabilities to show how the specific projects and program elements help to achieve a capability.
SvcV-1 Services Context Description	The identification of services, service items, and their interconnections.
SvcV-2 Services Resource Flow Description	A description of resource flows exchanged between services.
SvcV-3a Systems-Services Matrix	The relationships among or between systems and services in a given Architectural Description.
SvcV-3b Services-Services Matrix	The relationships among services in a given Architectural Description. It can be designed to show relationships of interest, (e.g., service-type interfaces, planned vs. existing interfaces).
SvcV-4 Services Functionality Description	The functions performed by services and the service data flows among service functions (activities)
SvcV-5 Operational Activity to Services Traceability Matrix	A mapping of services (activities) back to operational activities (activities).
SvcV-6 Services Resource Flow Matrix	It provides details of service resource flow elements being exchanged between services and the attributes of that exchange.
SvcV-7 Services Measures Matrix	The measures (metrics) of Services Model elements for the appropriate time frame(s).
SvcV-8 Services Evolution Description	The planned incremental steps toward migrating a suite of services to a more efficient suite or toward evolving current services to a future implementation.

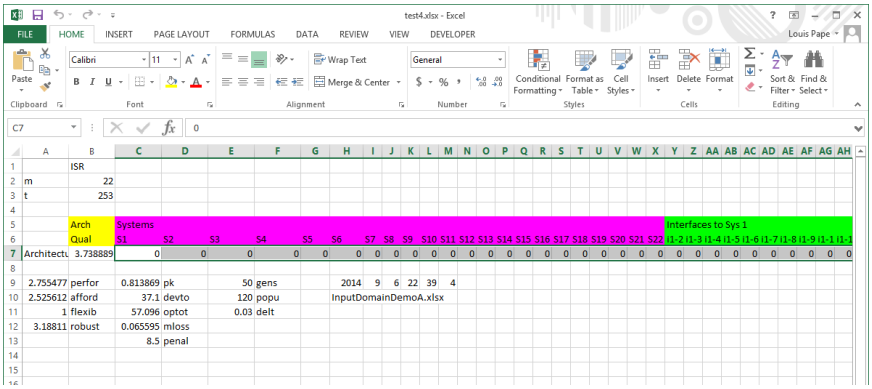
Models	Descriptions
SvcV-9 Services Technology & Skills Forecast	The emerging technologies, software/hardware products, and skills that are expected to be available in a given set of time frames and that will affect future service development.
SvcV-10a Services Rules Model	One of three models used to describe service functionality. It identifies constraints that are imposed on systems functionality due to some aspect of system design or implementation.
SvcV-10b Services State Transition Description	One of three models used to describe service functionality. It identifies responses of services to events.
SvcV-10c Services Event-Trace Description	One of three models used to describe service functionality. It identifies service-specific refinements of critical sequences of events described in the Operational Viewpoint.
StdV-1 Standards Profile	The listing of standards that apply to solution elements.
StdV-2 Standards Forecast	The description of emerging standards and potential impact on current solution elements, within a set of time frames.
SV-1 Systems Interface Description	The identification of systems, system items, and their interconnections.
SV-2 Systems Resource Flow Description	A description of resource flows exchanged between systems.
SV-3 Systems-Systems Matrix	The relationships among systems in a given Architectural Description. It can be designed to show relationships of interest, (e.g., system-type interfaces, planned vs. existing interfaces).
SV-4 Systems Functionality Description	The functions (activities) performed by systems and the system data flows among system functions (activities).
SV-5a Operational Activity to Systems Function Traceability Matrix	A mapping of system functions (activities) back to operational activities (activities).
SV-5b Operational Activity to Systems Traceability Matrix	A mapping of systems back to capabilities or operational activities (activities).
SV-6 Systems Resource Flow Matrix	Provides details of system resource flow elements being exchanged between systems and the attributes of that exchange.
SV-7 Systems Measures Matrix	The measures (metrics) of Systems Model elements for the appropriate timeframe(s).
SV-8 Systems Evolution Description	The planned incremental steps toward migrating a suite of systems to a more efficient suite, or toward evolving a current system to a future implementation.
SV-9 Systems Technology & Skills Forecast	The emerging technologies, software/hardware products, and skills that are expected to be available in a given set of time frames and that will affect future system development.
SV-10a Systems Rules Model	One of three models used to describe system functionality. It identifies constraints that are imposed on systems functionality due to some aspect of system design or implementation.
SV-10b Systems State Transition Description	One of three models used to describe system functionality. It identifies responses of systems to events.
SV-10c Systems Event-Trace Description	One of three models used to describe system functionality. It identifies system-specific refinements of critical sequences of events described in the Operational Viewpoint.

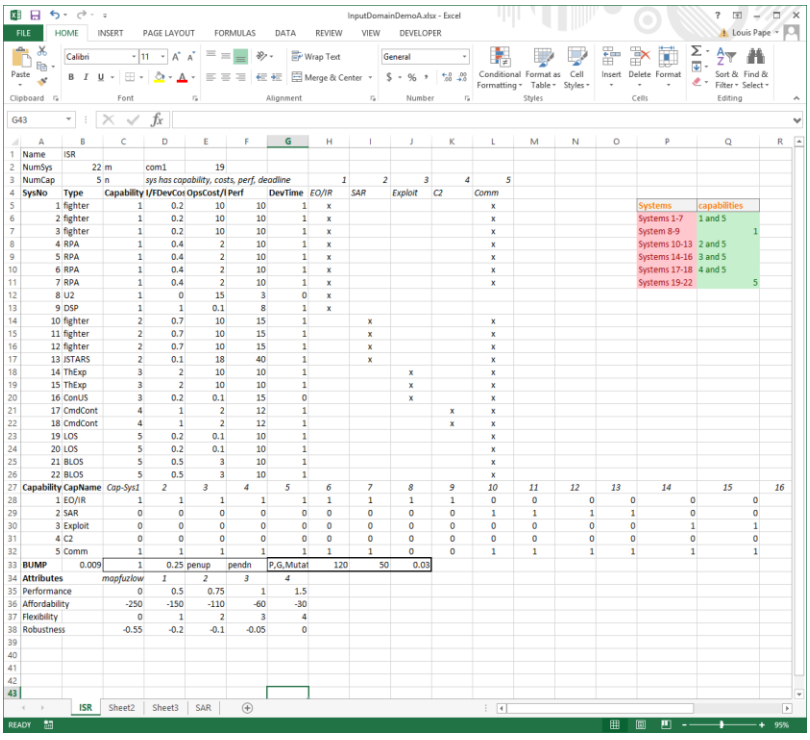
DoDAF Model	DoDAF Model Name	Typical Model Implementation
AV-1	Overview and Summary Information	Text (Word Document)
AV-2	Integrated Dictionary	Text, Spreadsheet or Database
OV-1	High Level Operational Concept Graphic	PowerPoint or Animator
OV-2	Operational Resource Flow Description	UML Collaboration Diagram
OV-3	Operational Resource Flow Matrix	Table, Spreadsheet or Database
OV-4	Organizational Relationships Chart	UML Class Diagram or Visio
OV-5b	Activity Model	UML Use Case Diagram, Sequence Diagram, Activity Diagram
OV-6c	Event Trace Description	UML Sequence or Activity Diagram
DIV-1	Conceptual Data Model	UML Classes & Class Diagrams
DIV-2	Logical Data Model	UML Classes & Class Diagrams
DIV-3	Physical Data Model	UML Classes & Class Diagrams
CV-1	Capability Vision	Text
CV-2	Capability Taxonomy	UML Class Diagram
CV-3	Capability Phasing	Table, Spreadsheet, Gantt Chart
CV-4	Capability Dependencies	UML Class Diagram
CV-5	Capability to Organizational Development Mapping	Table or Spreadsheet
CV-6	Capability to Organizational Activities Mapping	Table, Spreadsheet, Partitioned Activity Diagram, or Sequence Diagram
CV-7	Capability to Services Mapping	Table, Spreadsheet or UML Class Diagram
SvcV-1	Services Content Description	Text
SvcV-2	Services Resource Flow Description	UML Sequence Diagram
SvcV-3a	Systems-Services Matrix	Table or Spreadsheet
SvcV-3b	Services-Services Matrix	Table of Spreadsheet
SvcV-4	Services Functionality Description	Text
SvcV-5	Operational Activity to Services Traceability Matrix	Table or Spreadsheet
SvcV-6	Services Resources Flow Matrix	Table or Spreadsheet
SvcV-7	Services Measure Matrix	Table or Spreadsheet
SvcV-8	Services Evolution Description	Table, Spreadsheet, Gantt Chart
SvcV-9	Services Technology and Skills Forecast	Table, Spreadsheet, Gantt Chart
SvcV-10a	Services Rules Model	UML Activity Diagram
SvcV-10b	Services State Transition Description	UML State Diagram
SvcV-10c	Services Event-Trace Description	UML Sequence or Activity Diagram

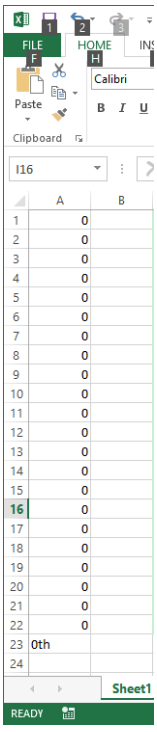
APPENDIX E

SUPPLEMENTARY FIGURES

The initial protection chromosome for the first wave is normally all zeroes – no selected systems or interfaces. The following screenshots show input and output files of the GA for the first wave:

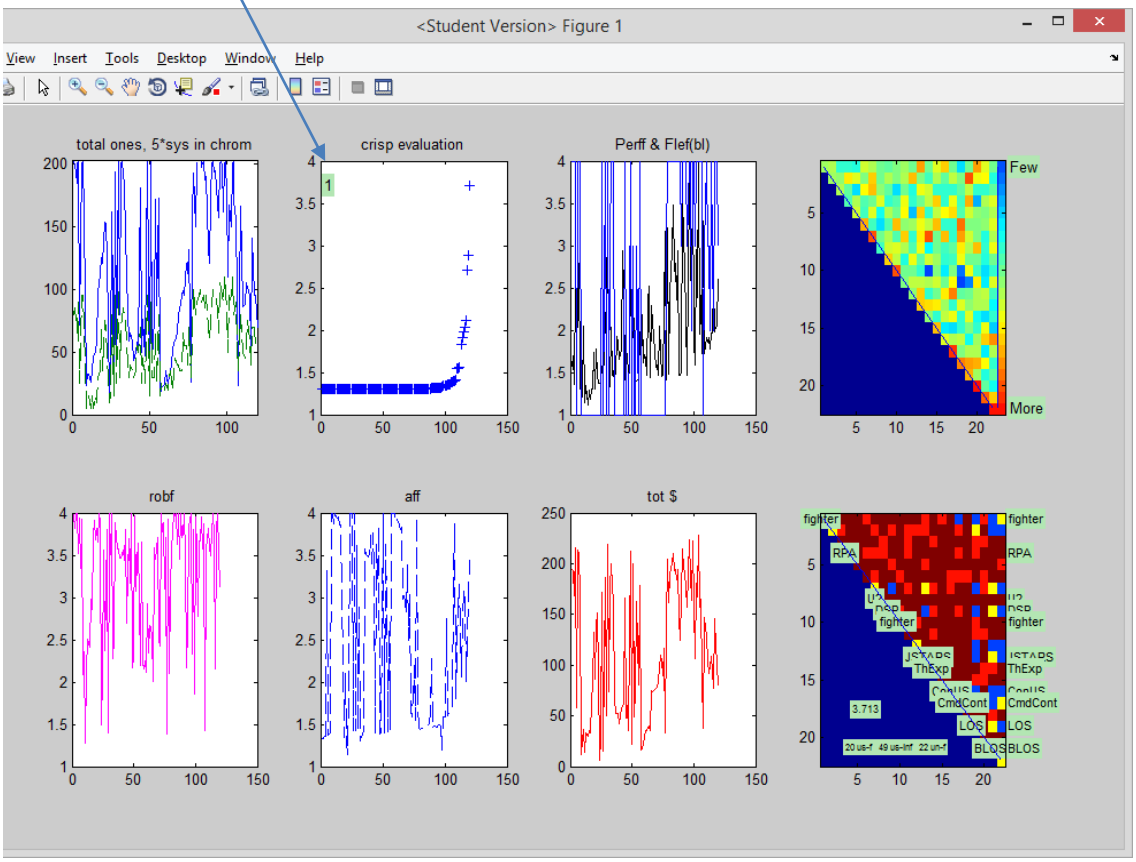


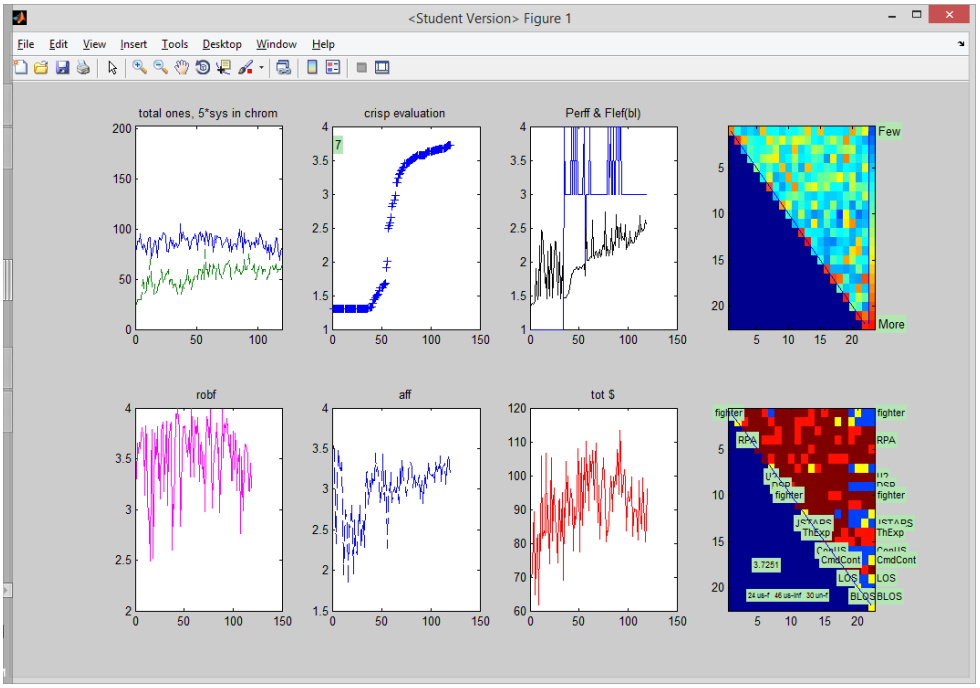


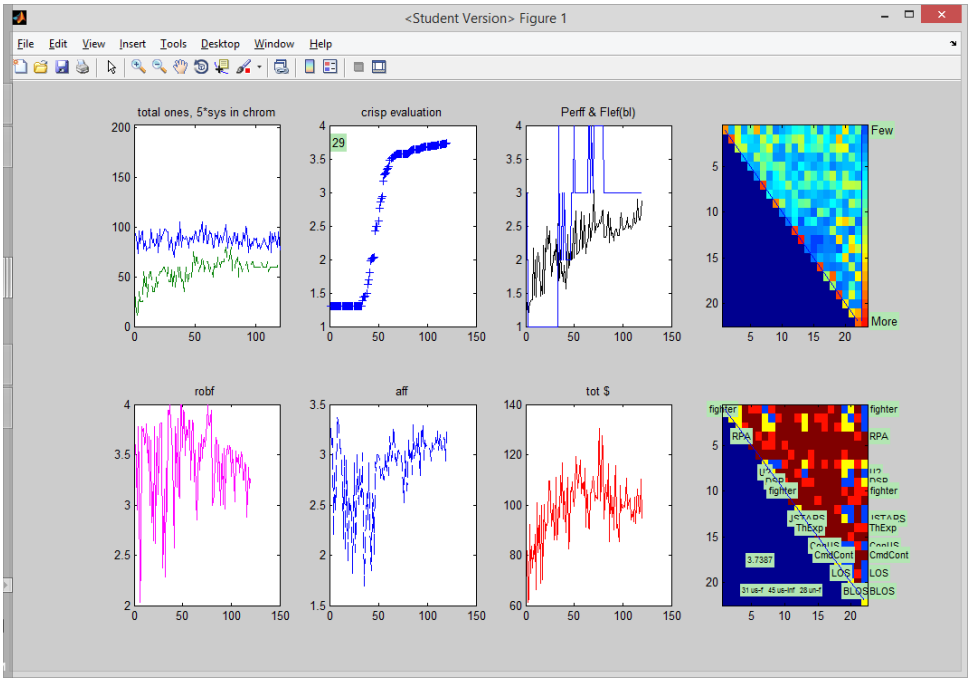


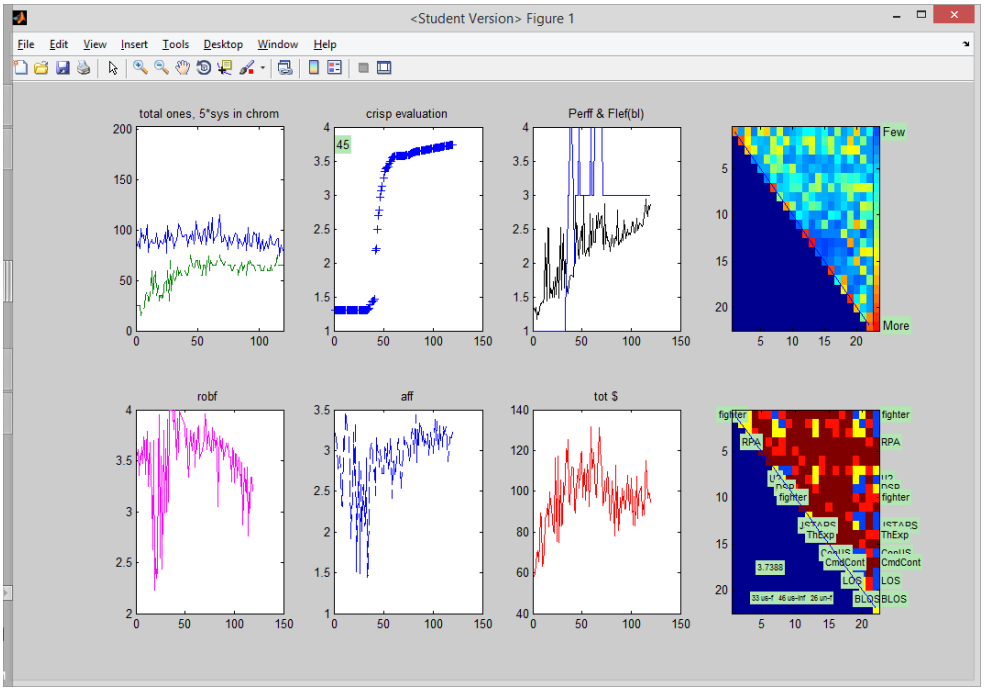
The following series of snapshots shows generations of the GA; note Generation 1 has a wide range of numbers of ones in both systems (5 times the number in red) and interfaces (blue)

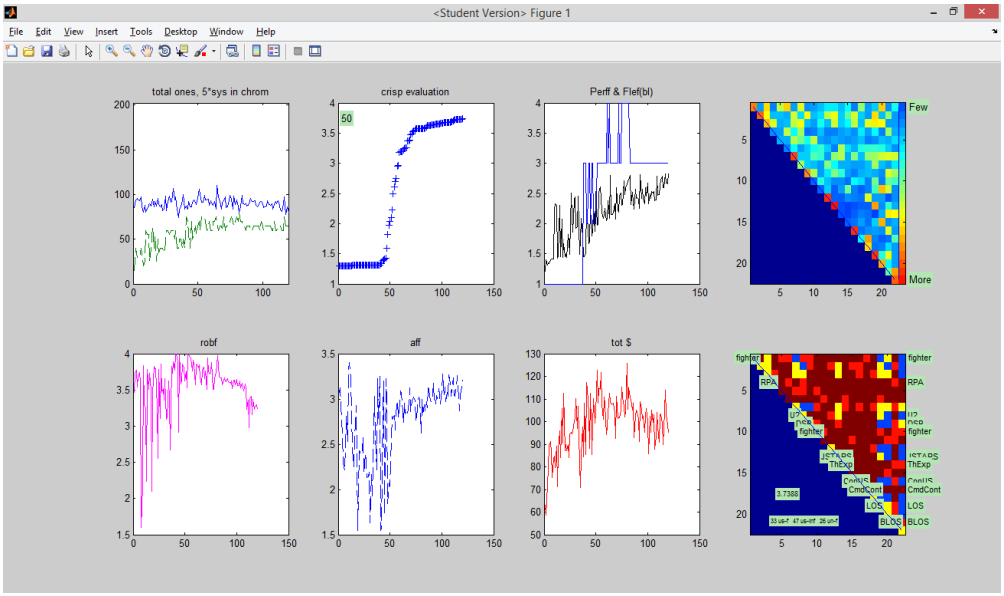
Generation number is in the upper left of the second graph.



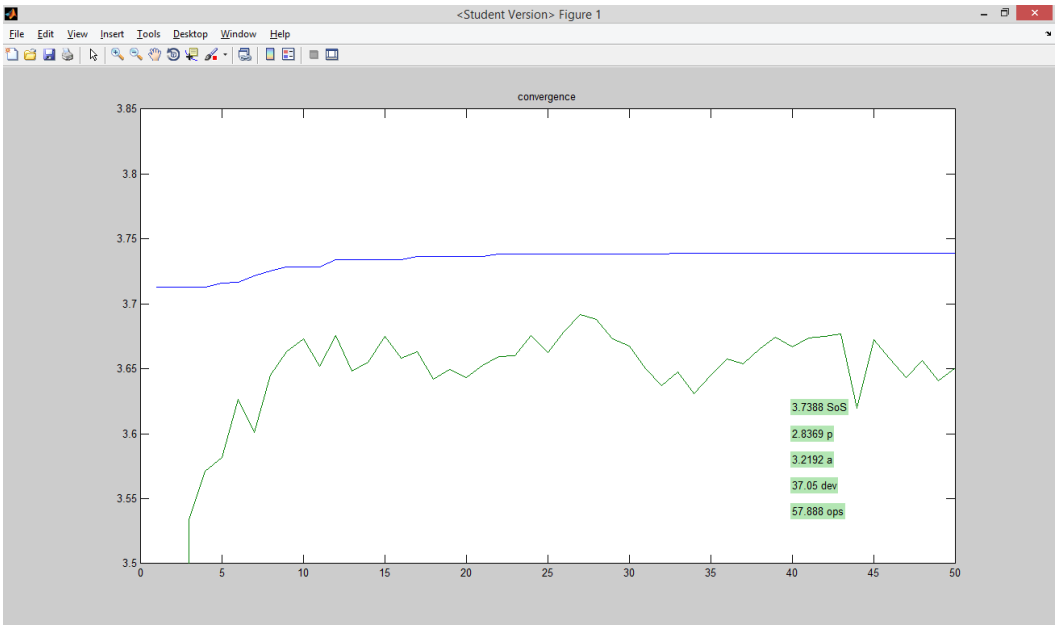

<Student Version> Figure 1
File Edit View Insert Tools Desktop Window Help
total ones, 5*sys in chrom
crisp evaluation
Perff & Flef(bl)
Few
More
robf
aff
tot $
fighter
RPA
ThExp
CmdCont
LOS
BLOS


<Student Version> Figure 1
File Edit View Insert Tools Desktop Window Help
total ones, 5*sys in chrom
crisp evaluation
Perff & Flef(bl)
Few
More
robf
aff
tot $
fighter
RPA
ThExp
CmdCont
LOS
BLOS

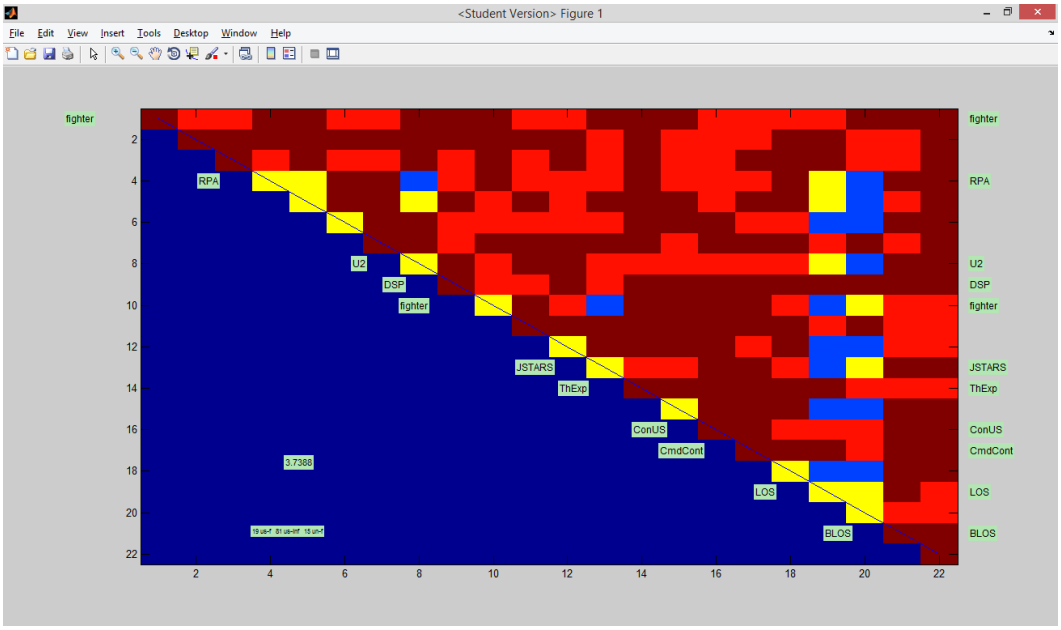




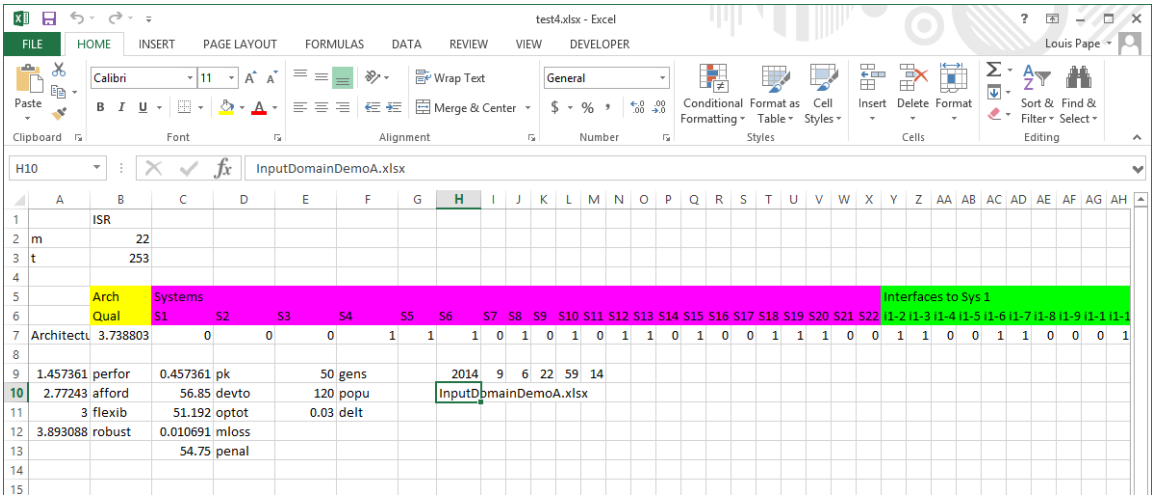
50th generation: the GA is complete



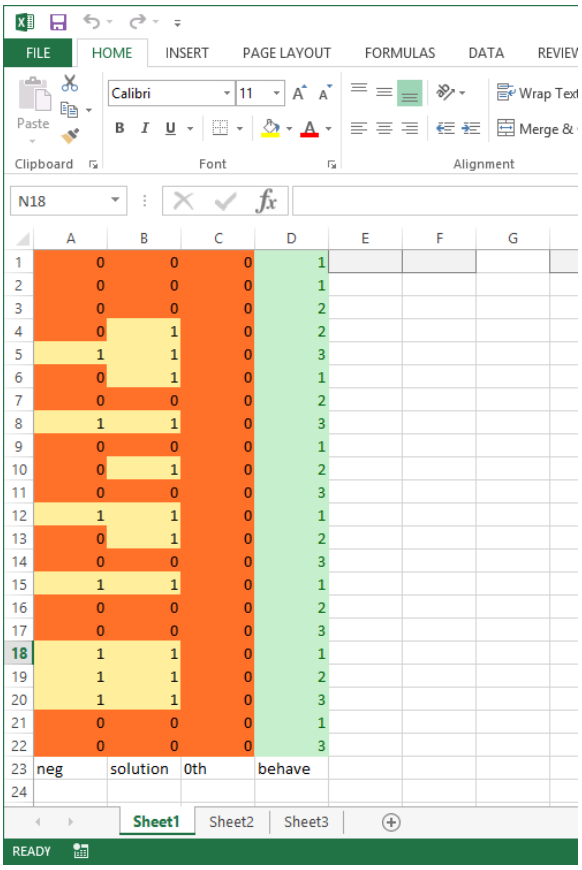
Convergence of best chromosome over generations (blue); the green line is the 20th population member



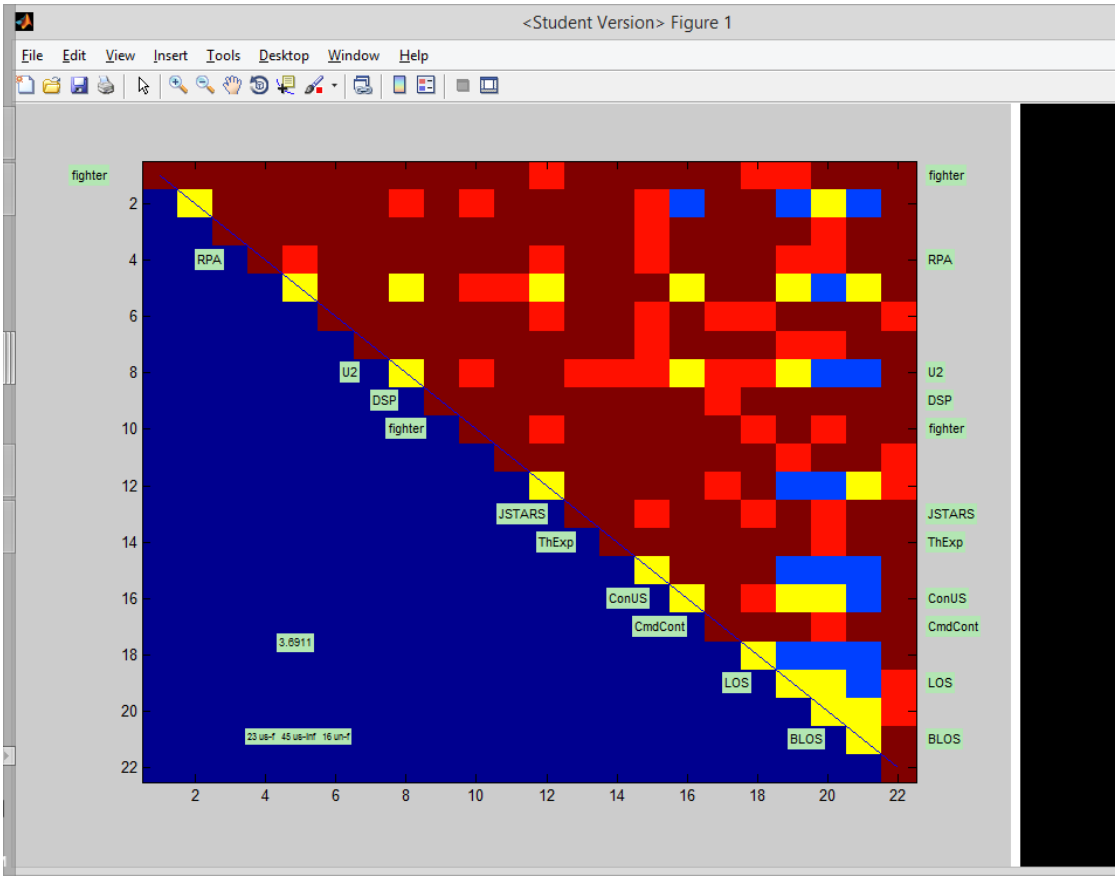
Final Upper Triangular Matrix representation



First wave chromosome in linear format:

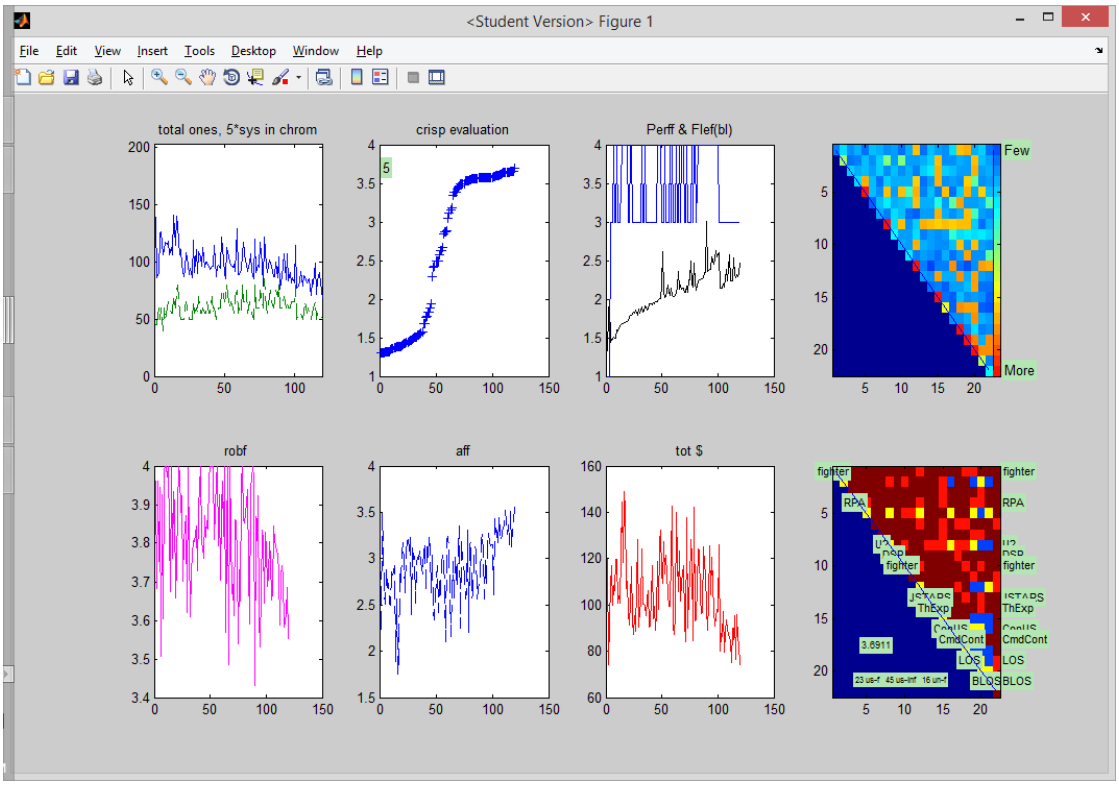


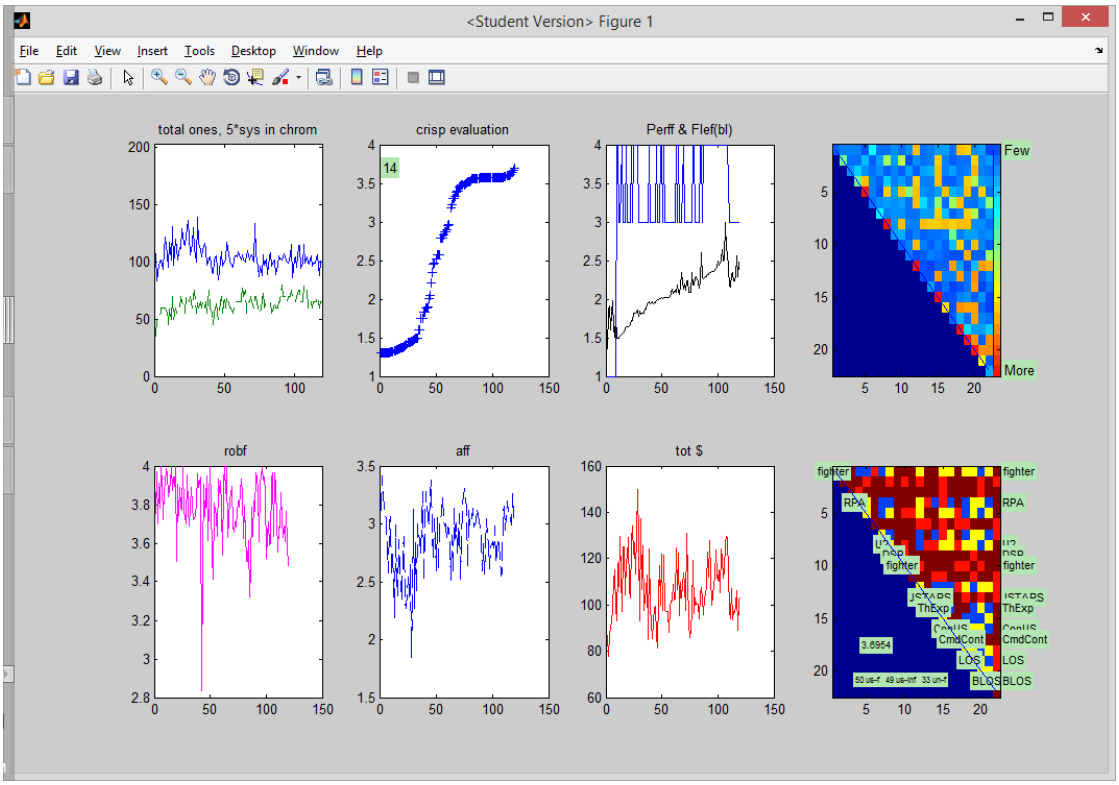
Showing solution of wave 1, after negotiating, a few less to start the second wave

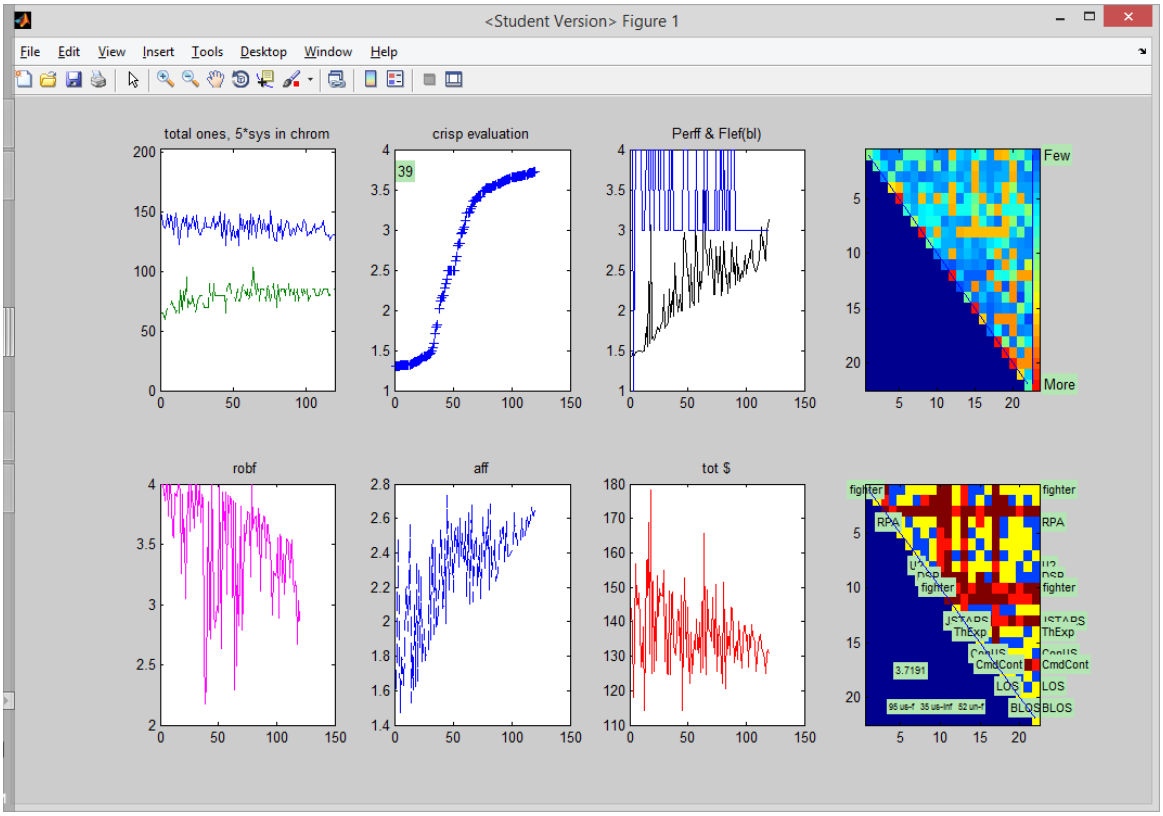


The final chromosome for this run of the GA

The following illustrations are of generations in the example second wave, wherein some systems and their previously negotiated interfaces are protected. This shows up as redder points in the heatmaps.

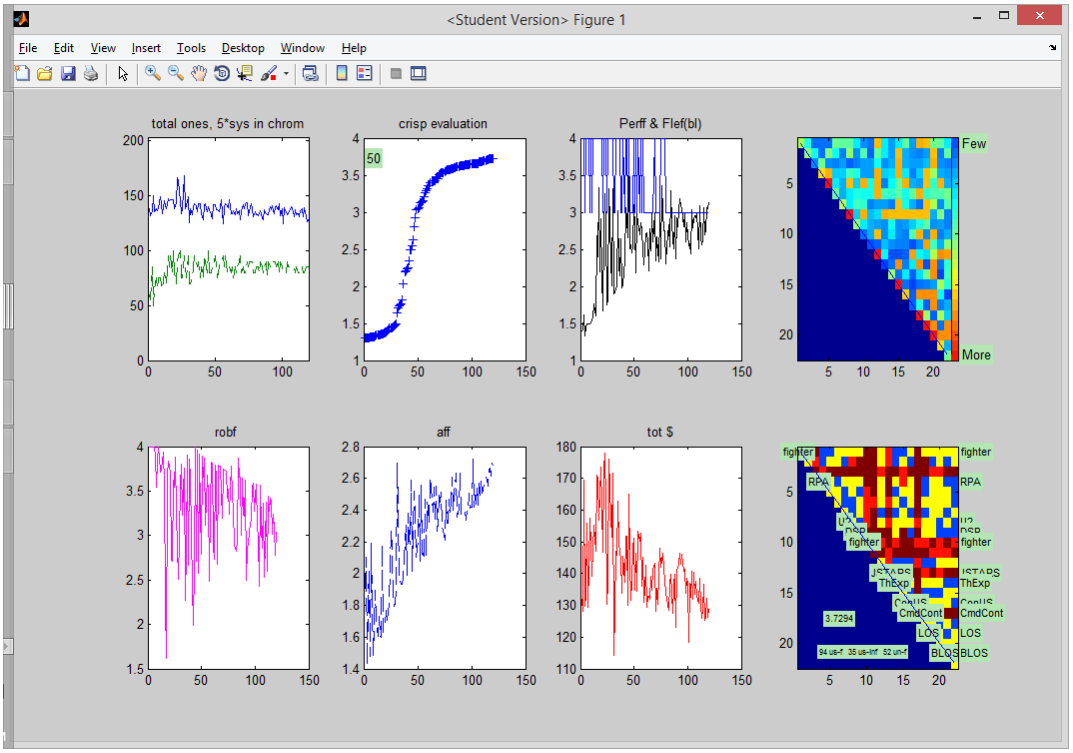

<Student Version> Figure 1
File Edit View Insert Tools Desktop Window Help
total ones, 5*sys in chrom
crisp evaluation
5
Perff & Flef(bl)
Few
More
robf
aff
tot $
fighter
RPA
ThExp
CmdCont
LOS
BLOS


<Student Version> Figure 1
File Edit View Insert Tools Desktop Window Help
total ones, 5*sys in chrom
crisp evaluation
14
Perff & Flef(bl)
Few
More
robf
aff
tot $
fighter
RPA
ThExp
CmdCont
LOS
BLOS

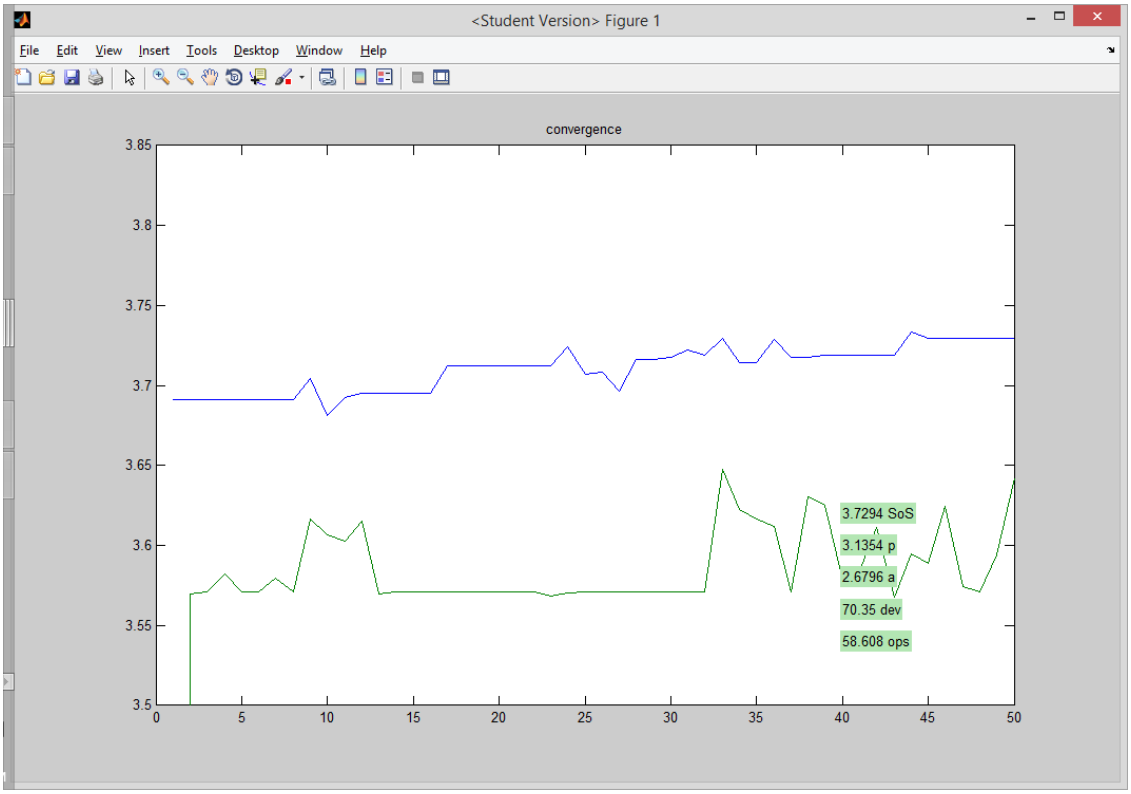


Remember: red on bottom chart is bad – unachievable but used; blue is 'could be better' – achievable but unused; green is best

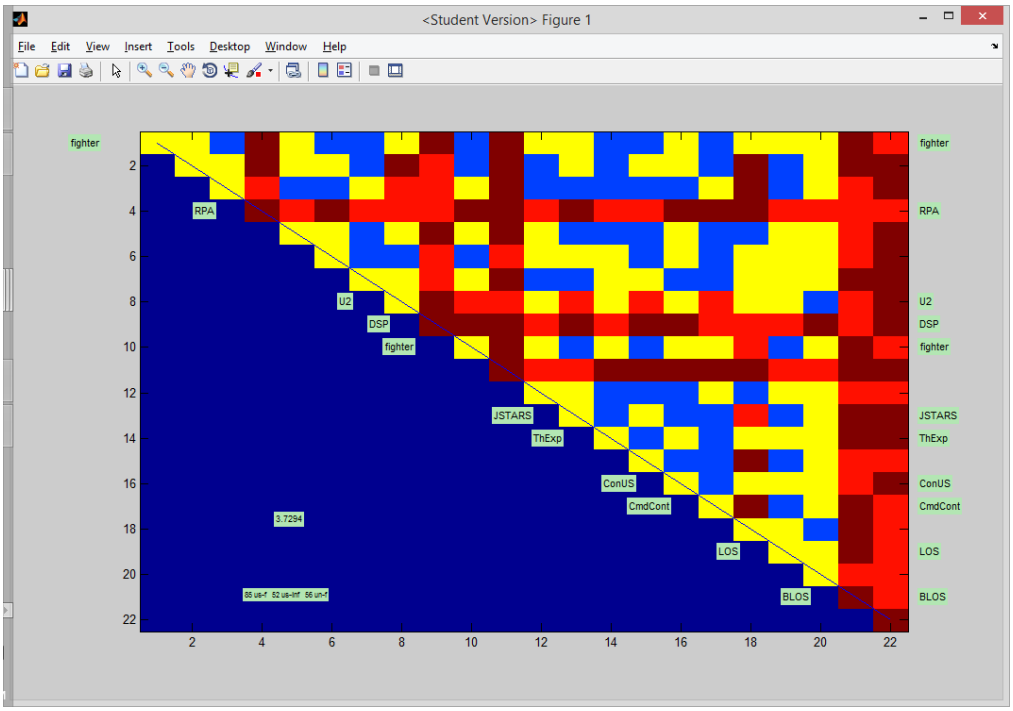
Heatmap color code (chart 4 on top row) is on the right side – from few (dark blue), to many (red)



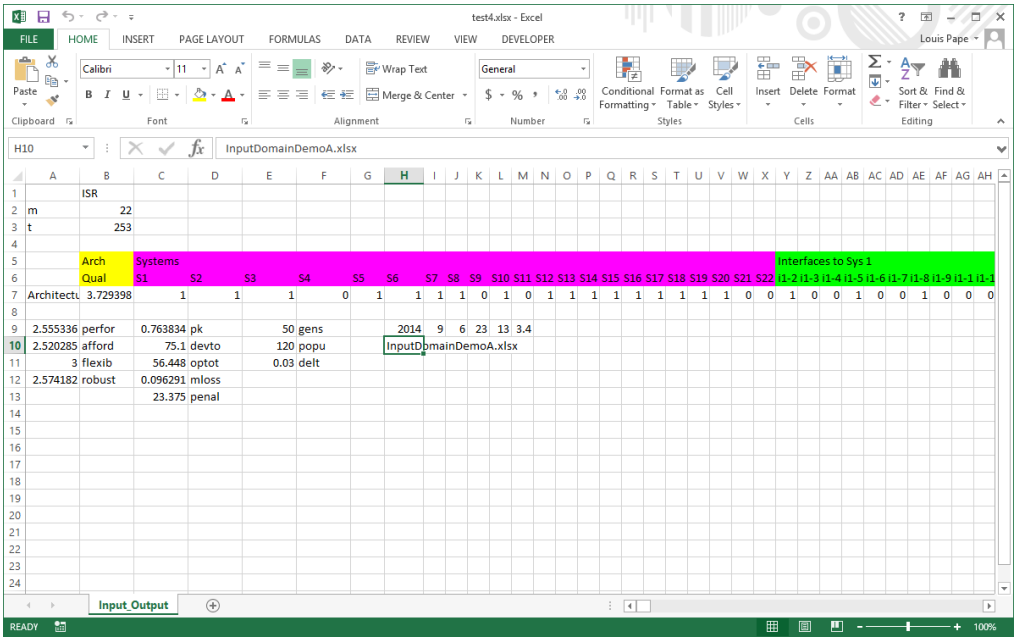
The heatmap shows red on a few interfaces, and all the systems that were marked as negotiated, as it should



Correcting mutations back to negotiated is occurring at the wrong place in this early version of the GA– although the trend is right, there should not be regressions in the blue convergence line. This was corrected at a later wave protection version of the code



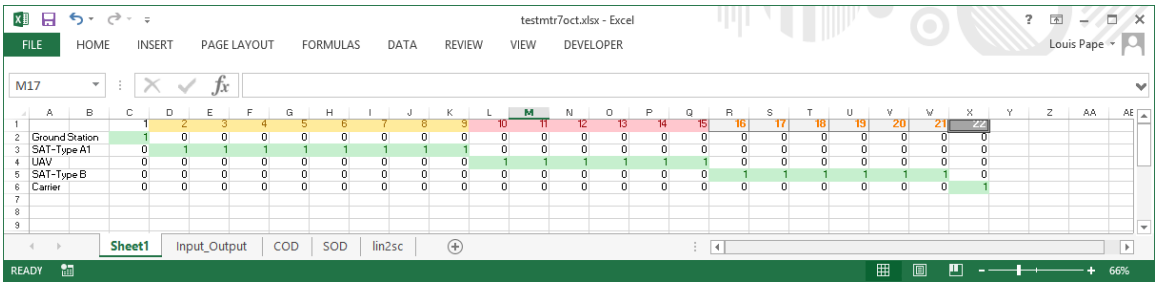
Second wave solution for ISR



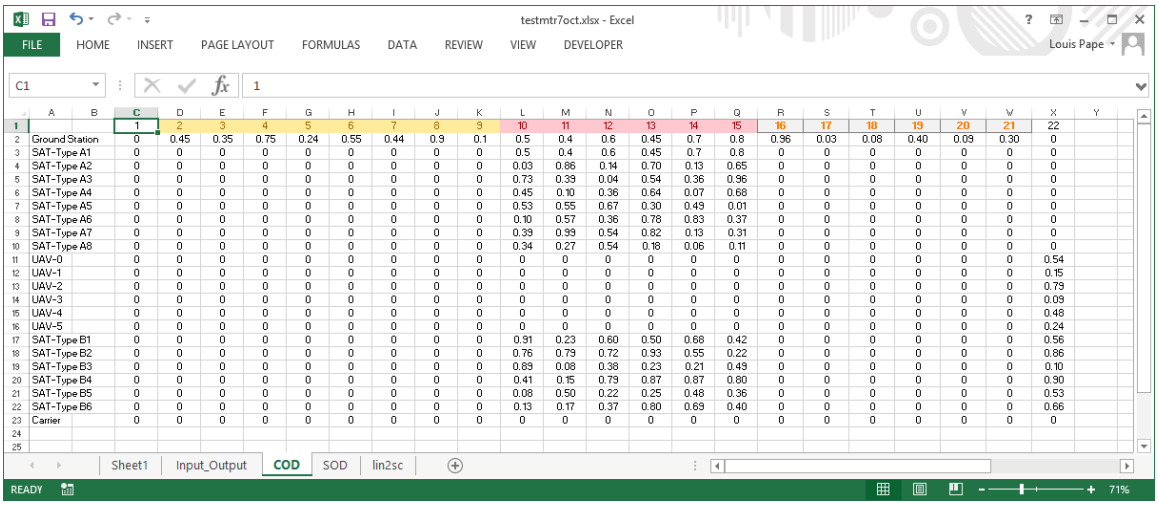
The Linear form of the chromosome shown immediately above

		Radius, Nautical miles	radius km	Speed, knots	Resolution, meters	Time on station, HOURS	Capabilities	Bandwidth	how controlled	Image size, m	flies at altitudes, feet	cost to modify for SoS interoperability	add'l cost to double the performance change on the left	cost to operate for a month (including personnel)	m2/sec	KM2/hr	pixels/sec (but these are compressed about 50 to 500 to one before transmission)
Raven	http://en.wikipedia.org/wiki/RQ-11_Raven	6.2	11.4824	30	0.15	1	EO or IR	2/sec stills	Controlled by soldier/operator	100	<500	can't modify it, not big enough, no networking, no recording, only connected to soldier; really:		$1,000	9000	32.4	1.20E+05
RQ-7 **Shadow**/Scan Eagle class	http://en.wikipedia.org/wiki/RQ-7_Shadow	59	109.268	80	0.2	9	EO or IR	1 Full Motion Video (FMV)	Controlled by command post	300-600	1000-5000	$10,000	$6,000	$60,000	24000	86.4	4.80E+06
MQ-1C **Gray Eagle**/Predator Class	http://en.wikipedia.org/wiki/MQ-1C_Gray_Eagle	200	370.4	120	0.1	25	EO/IR/SAR	2 FMV	Controlled distant remote	100-3000	15,000-25,000	$100,000	$150,000	$300,000	36000	129.6	1.44E+07
Apache Helicopter		200	370.4	180	0.2	1	EO/IR, strike	1 FMV, adjustable	piloted	3000	200-15000	$0	$500,000	$200,000	54000	194.4	1.08E+07
Command center surveillance desk	smaller exploitation capability than exp. Center	120	222.24	n/a	0.1	24/7	command, exploitation, fusion	8 voice, 3 FMV	command staff	100-3000	0	$30,000	$60,000	$90,000	#VALUE!	#VALUE!	#VALUE!
Control station (common)		120	222.24	n/a	0.1	24/7	coordination	2 voice, 1 FMV	controllers	100-3000	0	$5,000	$3,000	$250,000	#VALUE!	#VALUE!	#VALUE!
Exploitation Center		10000	18520	n/a	0.1	24/7	exploitation, fusion	infinite	analysts	100-5000	0	$0	$5,000	$100,000	#VALUE!	#VALUE!	#VALUE!
voice/chat		shared over either los or blos		n/a	n/a	24/7	coordination	3 KHz or 7 Kbs	anyone	n/a	n/a	$0	$0	$0	#VALUE!	#VALUE!	#VALUE!
LOS data comm		120	222.24	n/a	any	24/7	LOS	25-128 KBS	anyone	n/a	n/a	$30,000	$40,000	$0	#VALUE!	#VALUE!	#VALUE!
BLOS data comm		10000	18520	n/a	any	24/7	BLOS	32 KBS-10MBPS	RQ-7 goes through control station for BLOS	n/a	n/a	$100,000	$50,000	$15,000	#VALUE!	#VALUE!	#VALUE!
artillery (delivering shells from a 'battery')		25	46.3	1000	100	24/7	strike	n/a	manned	n/a	n/a	$10,000	$10,000	$200,000			

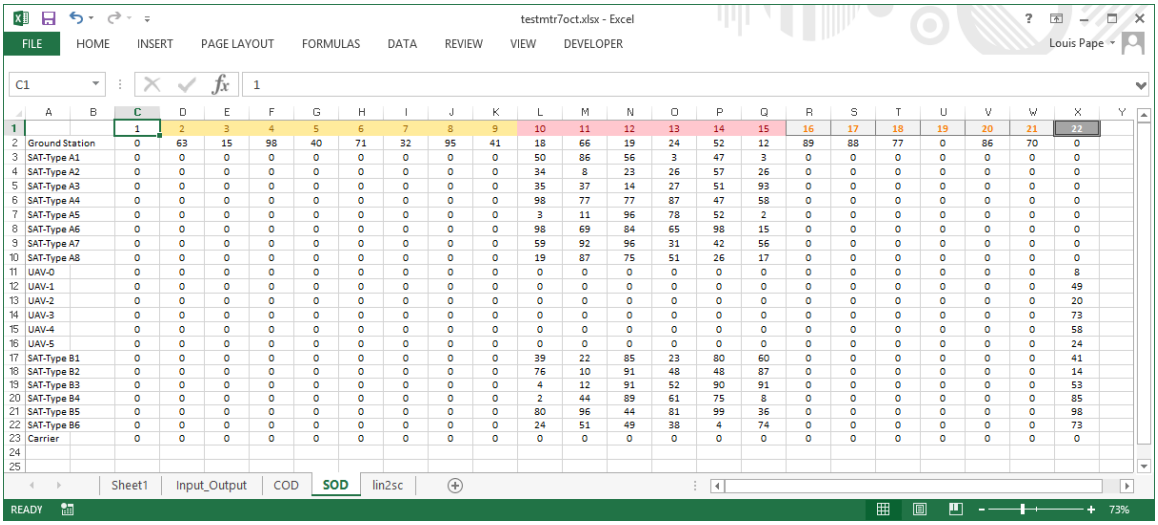
Input data sets are differently shaped for the Missouri Toy problem because a form of the FDNA evaluation is used for the performance attribute evaluation algorithm.



Capabilities of systems versus system type



Criticality of dependency to each system



testmtr7oct.xlsx - Excel

FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW DEVELOPER

Louis Pape

C1 | 1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
2	Ground Station		0	63	15	98	40	71	32	95	41	18	66	19	24	52	12	89	88	77	0	86	70	0
3	SAT-Type A1		0	0	0	0	0	0	0	0	0	50	86	56	3	47	3	0	0	0	0	0	0	0
4	SAT-Type A2		0	0	0	0	0	0	0	0	0	34	8	23	26	57	26	0	0	0	0	0	0	0
5	SAT-Type A3		0	0	0	0	0	0	0	0	0	35	37	14	27	51	93	0	0	0	0	0	0	0
6	SAT-Type A4		0	0	0	0	0	0	0	0	0	98	77	77	87	47	58	0	0	0	0	0	0	0
7	SAT-Type A5		0	0	0	0	0	0	0	0	0	3	11	96	78	52	2	0	0	0	0	0	0	0
8	SAT-Type A6		0	0	0	0	0	0	0	0	0	98	69	84	65	98	15	0	0	0	0	0	0	0
9	SAT-Type A7		0	0	0	0	0	0	0	0	0	59	92	96	31	42	56	0	0	0	0	0	0	0
10	SAT-Type A8		0	0	0	0	0	0	0	0	0	19	87	75	51	26	17	0	0	0	0	0	0	0
11	UAV-0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
12	UAV-1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	49
13	UAV-2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20
14	UAV-3		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	73
15	UAV-4		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	58
16	UAV-5		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24
17	SAT-Type B1		0	0	0	0	0	0	0	0	0	39	22	85	23	80	60	0	0	0	0	0	0	41
18	SAT-Type B2		0	0	0	0	0	0	0	0	0	76	10	91	48	48	87	0	0	0	0	0	0	14
19	SAT-Type B3		0	0	0	0	0	0	0	0	0	4	12	91	52	90	91	0	0	0	0	0	0	53
20	SAT-Type B4		0	0	0	0	0	0	0	0	0	2	44	89	61	75	8	0	0	0	0	0	0	85
21	SAT-Type B5		0	0	0	0	0	0	0	0	0	80	96	44	81	99	36	0	0	0	0	0	0	98
22	SAT-Type B6		0	0	0	0	0	0	0	0	0	24	51	49	38	4	74	0	0	0	0	0	0	73
23	Carrier		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24																								
25																								

Sheet1 | Input_Output | COD | SOD | lin2sc

READY 73%

Strength of dependency of links to each system

BIBLIOGRAPHY

j7jcaa@js.pentagon.mil. 2009. *Joint Capability Areas.* Washington DC, Jan 12.

Acheson, Paulette, Louis Pape, Cihan Dagli, Nil Kilcay-Ergin, John Columbi, and Khaled Haris. 2012. "Understanding System of Systems Development Using an Agent-Based Wave Model." *Complex Adaptive Systems, Publication 2.* Washington D.C.: Procedia Computer Science. 21-30.

Agarwal, Siddhartha, Louis E. Pape, and Cihan H. Dagli. 2014. "GA and PSO Hybrid with Type-2 Fuzzy Sets for Generating Systems of Systems Architectures." *Procedia Computer science.*

Ahn, Jae-Hong. 2012. "An Architecture Description method for Acknowledged System of Systems based on Federated Architecture." *Advanced Science and Technology Letters 05.*

Alberts, David S. 2011. *The Agility Advantage: A Survival Guide for Complex Enterprises and Endeavors.* Washington DC: Center for Advanced Concepts and Technology.

Alberts, David S., and Richard E. Hayes. 2005. *Power to the Edge: Command and Control in the Information Age.* Washington DC: Command and Control Research Program (CCRP).

Alberts, David S., John J. Garstka, and Frederick P. Stein. 1999. *Network Centric Warfare: Developing and Leveraging Information Superiority, 2nd Edition.* Washington DC: C4ISR Cooperative Research Program.

Arnold, A., B. Boyer, and A. Legay. 2012. "Contracts and Behavioral Patterns for System of Systems: The EU IP DANSE Approach."

ASD(NII), DoD. 2010. *DoD Architecture Framework Version 2.02 (DoDAF V2.02).* Washington DC: Department of Defense.

ASN/RDA. 2009. *Net-Ready Key Performance Parameter (NR-KPP) Implementation Guidebook, Ver. 1.* Guidebook, Washington DC: Department of the Navy.

Bergey, John K., Stephen Blanchette Jr, Paul C. Clements, Michael J. Gagliardi, John Klein, Rob Wojcik, and William Wood. 2009. "US Army Workshop on Exploring Enterprise, System of Systems, System, and Software Architectures."

Blanchard , Benjamin S., and Wolter J. Fabrycky . 2010. *Systems Engineering and Analysis.* Upper Saddle River, NJ: Prentice Hall.

Blekhman, Alex, and Dov Dori. 2011. "Model-Based Requirements Authoring." *4th Israeli International Conference on Systems Engineering.* Herzlia : INCOSE.

Brooks, Frederick P. 2010. *The Design of Design: Essays from a Computer Scientist.* Upper Saddle River, NJ: Addison-Wesley.

Cara, Ana Belen, Christian Wagner, Hani Hagras, Hector Pomares, and Ignacio Rojas. 2013. "Multiobjective Optimization and comparison of Nonsingleton Type-1 and Singleton Interval Type-2 Fuzzy Logic Systems." *IEEE Transactions on Fuzzy Systems* 21 (3): 459-476.

Christian III, John A. 2004. *A Quantitative Approach to Assessing System Evolvability.* Houston: NASA Johnson Space Center.

CJCSI 6212.01F. 12 Mar 2012. *Net Ready Key Performance Parameter (NR KPP).* Washington DC: US Dept of Defense.

Cloutier, Robert J., Michael J. Dimario, and Hans W. Polzer. 2009. "Net Centricity and System of Systems." In *System of Systems Engineering*, by Mo Jamshidi, 150-168. Hoboken NJ: John Wiley & Sons.

Clune, Jeff, Jean-Baptiste Mouret, and Hod Lipson. 2013. "The evolutionary origins of modularity." *Proceedings of the Royal Society - B* 280: 2863.

Coleman, Joey W., Anders Kaels Malmos, Peter gorm Larsen, Jan Peleska, Ralph Hains, Zoe Andrews, Richard Payne, et al. 2012. "COMPASS Tool Vision for a System of Systems Collaborative Development Environment." *Proceedings of the 7th International Conference on System of System Engineering, IEEE SoSE 2012.*

COMPASS. 2015. *COMPASS.* Mar 14. http://www.compass-research.eu/ .

Contag, G., C. Laing, J. Pabon, E. Rosenberg, K. Tomasino, and J. Tonello. 2013. *Nighthawk System Search and Rescue (SAR) Unmanned Vehicle (UV) System Development.* SE4150 Design Project, Naval Postgraduate School.

Dagli, Cihan, Nil Ergin, David Enke, Kristin Giammarco, Abhijit Gosavi, Ruwen Qin, Dincer Konur, et al. 2013. *An Advanced Computational Approach to System of Systems Analysis & Architecting using Agent Based Behavioral Modelin.* Final Technical Report, Systems Engineering Research Center, Hoboken NJ: SERC.

Dahmann, J., G. Rebovich, J. A. Lane, R Lowry, and K. Baldwin. 2011. "An Implementers' View of Systems Engineering for Systems of Systems." *Proceedings of IEEE International Systems Conference.* Montreal.

Dahmann, Judith. 2014. "System of System Pain Points." *INCOSE International Symposium.* Las Vegas: INCOSE.

Dahmann, Judith, Kristen J. Baldwin, and George Rebovich. 2009. "Systems of Systems and Net-Centric Enterprise Systems." *7th Annual Conference on Systems Engineering Research.* Loughborough.

Dam, Steven H., and Warren K. Vaneman,. 2015. *A New Open Standard: Lifecycle Modeling Language (LML) a Language for Simple, Rapid Development, Operations and Support.* March 12. http://cdn2.hubspot.net/hub/316256/file-493267217-pdf/LML_Overview_for_Lifecycle_Management_WG-Dam_and_Vaneman.pdf?t=1391103350000.

Dauby, Jason P. 2011. *ASSESSING SYSTEM ARCHITECTURES: THE CANONICAL DECOMPOSITION.* Rolla MO: Missouri University of Science & Technology.

De, P. K., and Bharti Yadav. 2011. "An Algorithm to Solve Multi-Objective Assignment Problem Using Interactive Fuzzy Goal Programming Approach." *International Journal of Contemporary Mathematical Sciences* 1651-1662.

Deb, Kalyanmoy, and Himanshu Gupta. 2006. "Introducing Robustness in Multi-Objective Optimization." *Evolutionary Computation* 14 (4): 463-494.

DeLaurentis, Daniel, Karen Marais, Navindran Davendralingam, Seung Yeob Han, Payuna Uday, Zhemei Fang, and Cesare Gurainiello. 2012. *Assessing the Impact of Development Disruptions and Dependencies in Analysis of Alternatives of System-of-Systems.* Final Technical Report SERC-2012-TR-035, Hoboken NJ: Stevens Institute of Technology, Systems Engineering Research Center.

Department of the Navy. 1997. "Contractor Performance Assessment Reporting System (CPARS)." Washington DC.

Deputy Minister of National Defence, and Commissioner, Canadian Coast Guard. 1998. *NATIONAL SEARCH AND RESCUE MANUAL (NATIONAL SAR MANUAL) B–GA–209–001/FP–001– DFO 5449.* Canadian Government.

Director Systems and Software Engineering, OUSD (AT&L). 2008. *Systems Engineering Guide for Systems of Systems.* available from http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf.

DYMASOS. 2015. *DYMASOS – Dynamic Management of Physically Coupled Systems of Systems.* March 12. http://www.dymasos.eu/outcomes/publications/dymasos-dynamic-management-of-physically-coupled-systems-of-systems/.

Eggstaff, Justin W., Thomas A. Mazzuchi, and Shahram Sarkani. 2014. "The Development of Progress Plans Using a Performance-Based Expert Judgment Model to Assess Technical Performance and Risk." *Systems Engineering* 375–391.

EUROCONTROL - The European Organisation for the Safety of Air Navigation. 2015. *eATM Portal; European ATM Master Plan.* March 6. https://www.atmmasterplan.eu/ .

European Commission of Transport. 2015. *European Commission of Mobility and Transport.* Mar 6. http://ec.europa.eu/transport/modes/air/single_european_sky/index_en.htm.

European Commission's FP7. 2015. *AMADEOS: Architecture for Multi-criticality Agile Dependable Evolutionary Open System-of-Systems.* March 14. http://amadeos-project.eu/.

Federal Aviation Administration. 2014. *NextGen Priorities Joint Implementation Plan: Executive Report to Congress.* Washington DC: FAA.

—. 2015. *NextGen Programs.* Accessed May 20, 2015. https://www.faa.gov/nextgen/programs/.

Flanagan, David, and Peggy Brouse. 2012. "System of Systems Requirements Capacity Allocation." *Procedia Computer Science* 8: 112-117.

Fogel, D.B. 2006. *Evolutionary Computation.* New Jersey: John Wiley and Sons.

Fry, Donald N., and Daniel A. DeLaurentis. 2011. "Measuring Net-Centricity." *Proceedings of the 6th International Conference on System of Systems Engineering.* Albuquerque.

Gao, Jianxi, Sergey V. Buldyrev, H. Eugene Stanley, and Shlomo Havlin. 2011. "Networks formed from interdependent networks." *Nature Physics* (Macmillan Publishers Ltd.) 8: 40-48. doi:10.1038/NPHYS2180.

Garvey, Paul R, and C. Ariel Pinto. 2009. "Introduction to functional dependency network analysis." *Second International Symposium on Engineering Systems.* Cambridge MA: MIT.

Gegov, Alexander. 2010. *Fuzzy Networks for Complex Systems.* Berlin: Springer.

Giachetti, Ronald E. 2012. "A Flexible Approach to Realize an Enterprise Architecture." *Procedia Computer Science* 8: 147-152.

Golkar, Alessandro, and Edward F. Crawley. 2014. "A Framework for Space Systems Architecture under Stakeholder Objectives Ambiguity." *Systems Engineering* 479–502.

Group, Asia/Pacific Seamless ATM Planning. 2013. *Asia/Pacific Seamless ATM Plan.* Bangkok: ICAO Asia and Pacific Office.

Guariniello, Cesare , and Daniel DeLaurentis . 2014. "Communications, information, and cyber security in Systems-of-Systems: Assessing the impact of attacks through interdependency analysis." *Procedia Computer Science; Conference on Systems Engineering Research.* Redondo Beach CA: Elsevier.

Haimes, Yacov Y. 2012. "Modeling complex systems of systems with Phantom System Models." *Systems Engineering* 333-346.

Haimes, Yacov Y., and Alfred Anderegg. 2015. "Sequential Pareto-Optimal Decisions Made During Emergent Complex Systems of Systems: An Application to the FAA NextGen." *systems Engineering* 28–44.

Han, Seung Yeob, and Daniel DeLaurentis. 2013. "Development Interdependency Modeling for System-of-Systems (SoS) using Bayesian Networks: SoS Management Strategy Planning." *Procedia Computer Science, Volume 16, 698–707.* Atlanta.

Hunt, B., R. L. Lipsman, and J. M. Rosenberg. 2001. *A guide to MATLAB: For beginners and experienced users.* New York: Cambridge University Press.

IEEE S2ESC – Software and Systems Engineering Standards Committee). 2011. *ISO/IEC/IEEE 42010:2011, Systems and software engineering — Architecture description.* New York: Institute of Electrical and Electronics Engineers, Inc.

INCOSE. 2011. *Systems Engineering Handbook, v. 3.2.2.* INCOSE-TP-2003-002-3.2.2, San Diego: INCOSE.

ISO/IEC/IEEE. 2008. *15288-2008 - ISO/IEC/IEEE Systems and Software Engineering — System Life Cycle Processes.* Geneva: ISO.

ISO/IEC/IEEE. 2011. *ISO/IEC/IEEE 42010:2011 Systems and software engineering -- Architecture description.* Geneva: ISO Org.

Jackson, Scott, and Timothy L. J. Ferris. 2013. "Resilience Principles for Engineered Systems." *Systems Engineering* 16 (2): 152-164.

Johnston, W., K. Mastran, N. Quijano, and M. Stevens. 2013. *Unmanned Vehicle Search and Rescue Initiative.* SE4150 Design Project, Naval Postgraduate School.

Joint Staff. 2010. *CJCSM 3500.04C, UNIVERSAL JOINT TASK LIST (UJTL).* Manual, Washington DC: Department of Defense.

Kinnunen, Matti J. 2006. *Complexity Measures for System Architecture Models.* Cambridge: MIT: System Design and Management Program Masters Thesis.

Kooistra, Rien L., G. Maarten Bonnema, and Jacek Sko. 2012. "A3 Architecture Overviews for Systems-of-Systems." *Complex Systems Design & Management (CSD&M).* Paris: Springer-Verlag.

Kruchten , Philippe. 1995. "The 4+1 View Model of Architecture." *IEEE Software* 45-50.

Krugman, Paul, and Robin Wells. 2009. *Economics.* New York: Worth Publishers.

Kujawski, Edouard. 2014. "Interaction Effects in the Design of Computer Simulation Experiments for Architecting Systems-of-Systems." *Systems Engineering* 426–441.

Kumar, Rakesh, and Jyotishree. 2012. "Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms." *International Journal of Machine Learning and Computing* 2 (4): 365-370.

Lafleur, Jarret M. 2012. *A Markovian State-SpaceFramework for Integrating Flexibility into Space System Design Decisions.* Atlanta: Georgia Institute of Technology School of Aerospace Engineering Doctoral Thesis.

Li, Chunshien, and Tai-Wei Chiang. 2013. "Complex Neurofuzzy ARIMA Forecasting - A New Approach Using Complex Fuzzy Sets." *IEEE Transactions on Fuzzy Systems* 21 (3): 567-584.

Lin, Yinghua, George A. Cunningham III, Stephen V. Coggshall, and Roger D. Jones. 1998. "Nonlinear System Input Structure Identification: Two Stage Fuzzy Curves and Surfaces." *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 28 (5): 678- 684.

Madni, Azad M., and Michael Sievers. 2014. "System of Systems Integration: Key Considerations and Challenges." *Systems Engineering* 330–347.

Maier, Mark W., and Eberhardt Rechtin. 2009. *The Art of Systems Architecting, 3rd ed.* Boca Raton: CRC Press.

Mekdeci, Brian, Nirav Shah, Adam M. Ross, Donna H. Rhodes, and Daniel Hastings. 2014. *Revisiting the Question: Are Systems of Systems just (traditional) Systems or are they a new class of Systems?* CESUN Conference, Cambridge, MA: Systems Engineering Advancement Research Initiative (SEAri).

Mendel, Jerry M. 2013. "On KM Algorithms for Solving Type-2 Fuzzy Set Problems." *IEEE Transactions on Fuzzy Systems* 21 (3): 426-446.

MONDO Project. 2015. *MONDO – Scalable Modeling and Model Management on the Cloud.* March 12. http://www.mondo-project.org/.

Mordecai, Yaniv, and Dov Dori. 2013. "I5: A Model-Based Framework for Architecting System-of-Systems Interoperability, Interconnectivity, Interfacing, Integration, and Interaction." *International Symposium of the International Council on Systems Engineering (INCOSE).* Philadelphia.

NASA. 2007. *Space Systems Engineering.* Accessed June 8, 2011. http://spacese.spacegrant.org/index.php?page=videos.

Nord, Robert L., Paul C. Clements, David Emery, and Rich Hilliard. 2009. *A Structured Approach for Reviewing Architecture Documentation.* Pittsburgh, PA: Carnegie Mellon University Software Engineering Institute.

Pape, Louis, and Cihan Dagli. 2013. "Assessing robustness in systems of systems meta-architectures." *Procedia Computer Science, Complex Adaptive Systems.* Baltimore: Elsevier. 262-269.

Pape, Louis, Kristin Giammarco, John Colombi, Cihan Dagli, Nil Kilicay-Ergin, and George Rebovich. 2013. "A fuzzy evaluation method for system of systems meta-architectures." *Procedia Computer Science, Conference on Systems Engineering Research (CSER'13).* Atlanta: ScienceDirect, Elsevier. 245-254.

Paulish, Daniel J., and Len Bass. 2001. *Architecture-Centric Software Project Management: A Practical Guide.* Boston: Addison-Wesley Longman Publishing Co., Inc.

Pedrycz, Witold, Petr Ekel, and Roberta Parreiras. 2011. *Fuzzy Multicriteria Decision Making; Models, Methods and Applications.* West Sussex: John Wiley & Sons.

Pitsko, Robert, and Dinesh Verma. 2012. "Principles for Architecting Adaptable Command and Control." *New Challenges in Systems Engineering and Architecting.* St. Louis.

Ricci, Nicola, Adam M. Ross, Donna H. Rhodes, and Matthew E. Fitzgerald. 2013. *Considering Alternative Strategies for Value Sustainment in Systems-of-Systems (Draft).* Cambridge MA: Systems Engineering Advancement Research Initiative.

Rosenau, William. 1991. *Coalition Scud Hunting in Iraq, 1991.* RAND Corporation.

Ross, Adam M. 2014. "Contributing toward a Prescriptive "Theory of Ilities"." *Systems Engineering Advancement Research Initiative (SEARri).* April 2. http://seari.mit.edu/documents/presentations/MST14_Ross_MIT.pdf.

Ross, Adam M., Donna H. Rhodes, and Daniel E. Hastings. 2008. "Defining Changeability: Reconciling Flexibility, Adaptability, Scalability, Modifiability, and Robustness for Maintaining System Lifecycle Value." *Systems Engineering* 246 - 262.

Rostker, Bernard. 2000. "Iraq's Scud Ballistic Missiles." *Iraq Watch.* July 25. Accessed Sep 12, 2013. http://www.iraqwatch.org/government/US/Pentagon/dodscud.htm .

Sanz, Jose Antonio, Alberto Fernandez, Humberto Bustince, and Francisco Herrara. 2013. "IVTURS: A Linguistic Fuzzy Rule-Based Classification System Based On a New Interval-Valued Fuzzy Reasoning Method with Tuning and Rule Selection." *IEEE Transactions on Fuzzy Systems* 21 (3): 399-411.

Schreiner, Michael W., and Joseph R. Wirthlin. 2012. "Challenges using modeling and simulation in architecture." *Procedia Computer Science* (ScienceDirect) 8: 153-158.

Selva, Daniel, and Edward F. Crawley. 2013. "VASSAR: Value Assessment of System Architectures using Rules." *IEEE Aerospace Conference.* Big Sky MT: IEEE. 1-21.

SESARJU. 2015. *SESAR Joint Undertaking.* Mar 7. http://www.sesarju.eu/ .

Shaw, Albert, ed. 1918. *President Wilson's State Papers and Addresses.* New York: George H. Doran Co.

Singer, Yariv. 2006. "Dynamic Measure of Network Robustness." *IEEE 24th Conference of Electrical and Electronic Engineers in Israel.*

Singh, Atmika. 2011. *Architecture Value Mapping: Using Fuzzy Cognitive Maps As A Reasoning Mechanism For Multi-Criteria Conceptual Design Evaluation. Dissertation.* Rolla MO: Missouri University of Science & Technology.

Singh, Atmika, and Cihan H. Dagli. 2010. ""Computing with words" to support multi-criteria decision-making during conceptual design." *Systems Research Forum* 85-99.

—. 2009. "Multi-objective Stochastic Heuristic Method for Trade Space Exploration of a Network Centric System of Systems." *3rd Annual IEEE International Systems Conference, 2009.* Vancouver, Canada.

Smartt, Clement, and Susan Ferreira. 2012. "Constructing a General Framework for Systems Engineering Strategy." *Systems Engineering* 15 (2): 140-152.

Software Engineering Institute, Carnegie Mellon University. 2015. *Active Reviews for Intermediate Design.* Accessed March 22, 2015. http://www.sei.cmu.edu/architecture/tools/evaluate/arid.cfm.

—. 2015. *Architecture Tradeoff Analysis Method.* March 12. http://www.sei.cmu.edu/architecture/tools/evaluate/atam.cfm.

—. 2015. *System of Systems Architecture Evaluation Method.* Mar 12. http://www.sei.cmu.edu/architecture/tools/evaluate/sosevaluation.cfm.

SPEC Innovations. 2015. *Model-Based Systems Engineering Tools.* March 18. Accessed August 20, 2014. https://www.innoslate.com/systems-engineering/.

Stephenson, Neal. 2011. "Innovation Starvation." *World Policy Journal* (Sage) 28 (3): 11-16.

Suarez, Ray. 2004. "Troops Question Secretary of Defense Donald Rumsfeld about Armor." *PBS NewsHour.* Dec 9. Accessed Apr 14, 2014. http://www.pbs.org/newshour/bb/military-july-dec04-armor_12-9/.

Sumathi, S., and P. Surekha. 2010. *Computational Intelligence Paradigms: Theory & Applications Using MATLAB.* Boca Raton FL: CRC Press.

Taleb, Nassim Nicholas. 2004. *Fooled by Randomness.* New York: Random House Trade Paperbacks.

Thompson, Mark. 2002. "Iraq: The Great Scud Hunt." *Time Magazine*, December 23: http://www.time.com/time/magazine/article/0,9171,1003916,00.html.

Trans-Atlantic Research and Education Agenda in Systems of Systems (T-AREA-SOS) Project. 2013. *The Systems of Systems Engineering Strategic Research Agenda.* Loughborough: Loughborough University.

Wai, Jonathan. 2012. "The Growing Complexity of Everyday Life." *Psychology Today*, Nov 12: 25.

Wang, Jian-Qiang, and Hong-Yu Zhang. 2013. "Multicriteria Decision-Making Approach Based on Atanassov's Intuitionistic Fuzzy Sets With Incomplete Certain Information on Weights." *IEEE Transactions on Fuzzy Systems* 21 (3): 510-515.

Warfield, John N. 1973. "Binary Matrices in System Modeling." *IEEE Transactions on Systems, Man, and Cybernetics* SMC-3 (No. 5, September): 441-449.

West, Douglas B. 2000. *Introduction to Graph Theory (2nd Ed.).* Urbana IL: University of Illinois.

Yi, Ji Soo, Youh ah Kang, John T. Stasko, and Julie A. Jacko. 2007. "Toward a Deeper Understanding of the Role of Interaction in Information Visualization." *IEEE Transactions on Visualization and Computer Graphics* 13 (6): 1224 - 1231.

Yu, O.-Y., S. D. Gulkema, J.-L Briaud, and D. Burnett. 2011. "Sensitivity Analysis for Multi-Attribute System Selection Problems in Onshore Environmentally Friendly Drilling (EFD)." *Systems Engineering* 15 (2): 153-171.

Zadeh, L. A. 1975. "Fuzzy logic and approximate reasoning." *Synthese* 30 (3-4): 407-428.

VITA

Louis E. Pape II was born in Chicago, Ilinois. In June 1970, he graduated from the US Air Force Academy as a USAF lieutenant with a BS in Physics. He earned an MS in Optical Sciences at the University of Arizona in February 1972. He conducted laser propagation and atmospheric turbulence experiments in wind tunnels and aircraft until 1976. He served at the Pentagon as Assistant Program Element Monitor (PEM) on the Global Positioning System (GPS), and Acting PEM for USAF Basic Research. He managed GPS Satellite contract finances for four years at Space and Missile Systems Organization. He left active duty in 1980, remaining in the AF Reserve for 20 years. He retired as a Colonel in 2000, having served in acquisition, logistics, training, testing and safety positions. After leaving the USAF he worked for TRW in Redondo Beach, CA on satellite laser communication systems. In 1994 he joined Boeing to work on aircraft systems engineering, and networked systems. In 2009 he became a Boeing Associate Technical Fellow in Systems Engineering. He was in instructor in the Boeing Systems Engineering Leadership Program starting in 2006.

He earned an MBA in 1980 from California State University, Dominguez Hills. He graduated from the Industrial College of the Armed Forces (1986), and Air War College (1996). In May 2016, he received his Ph.D. in Systems Engineering from the Missouri University of Science and Technology, Rolla, Missouri.

Louis E. Pape II was a member of IEEE, INCOSE (an officer in the Midwest Gateway chapter). He became a Certified Systems Engineering Professional (CSEP) in 2011. He has authored several conference papers and reports. In 2013 he was the winner of the INCOSE/Stevens Institute Doctoral Award for "research most likely to impact Systems Engineering in the next 10 years." He has mentored both FIRST Robotics teams and student project teams in the Missouri S&T *Introduction to SE* course for several years.