A Fuzzy Evaluation Method for System of Systems Meta-Architectures

Mr. Louis Pape, The Boeing Company

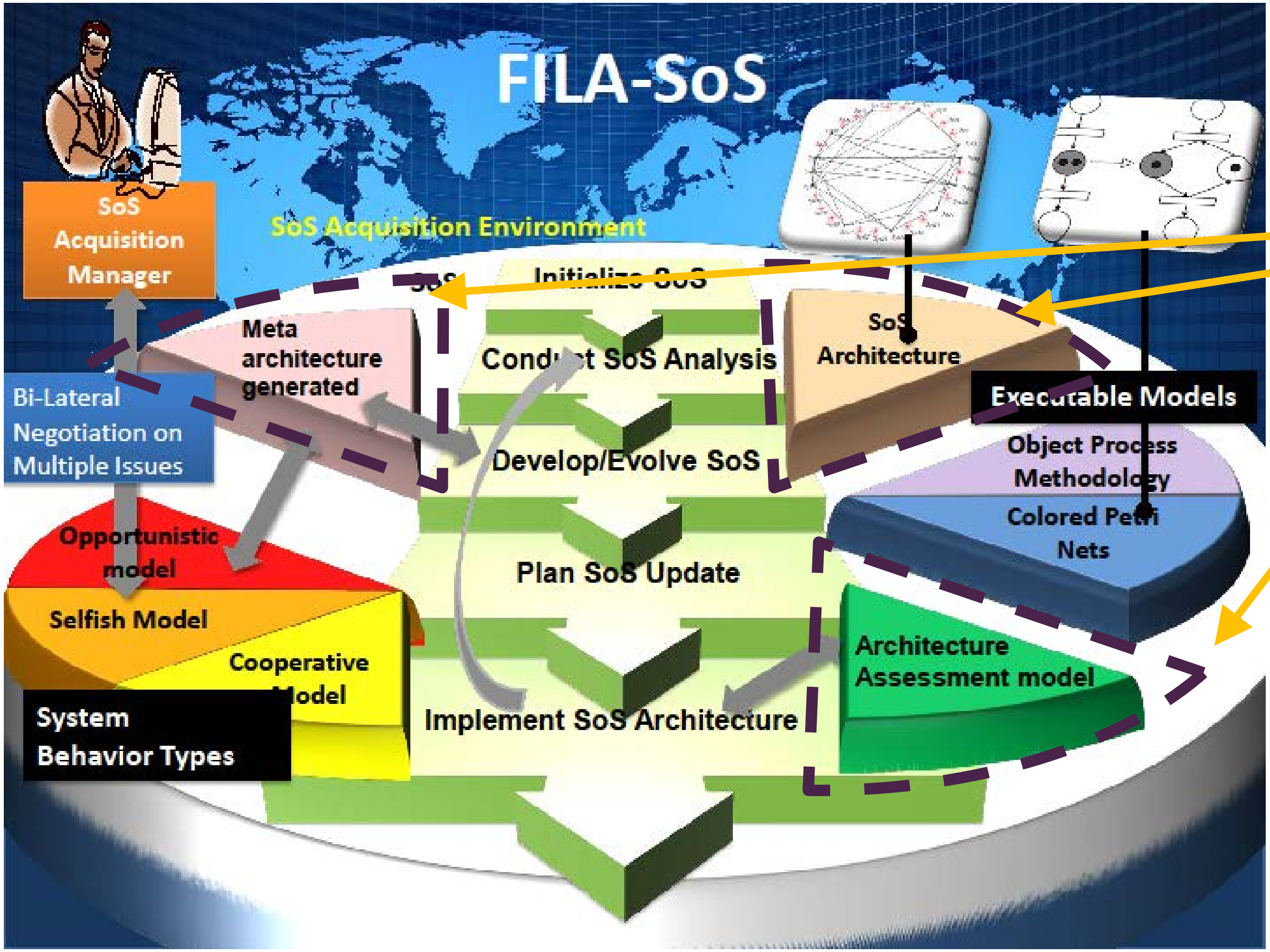
lep7df@mst.edu

4 November 2014

Agenda

- Acknowledged SoS and the wave model – FILA-SoS
- The meta-architecture & *binary* string representation
  - Input domain data
- Eliciting evaluation criteria (-ilities)
- Combining criteria to an overall SoS quality
- Fuzzy implementations & determination of SoS fitness
  - How the criteria/attributes depend on the architecture
- Genetic algorithm evaluation of alternatives
  - Non-linear twists and end-around checks
- Examples
- Lessons learned

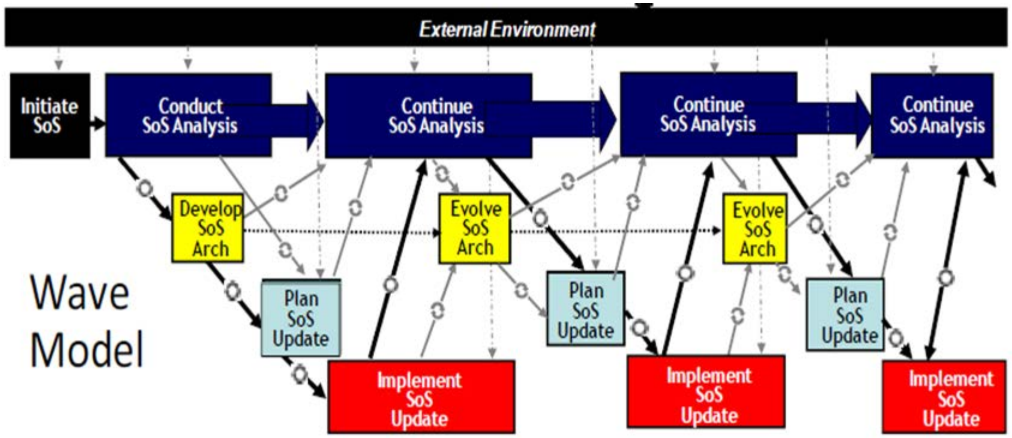
FILA-SoS: Flexible and Intelligent Learning Architectures for Systems of Systems



Focus of This Presentation

Acknowledged SoS and the Wave Model

- Acknowledged SoS are not commanded: they are a coalition of the willing
  - Existing missions are minimally impacted
  - Changes are kept minor
  - Budgets are relatively small
- Possibility of a quick but large improvement triggers an acknowledged SoS
- Expect SoS to improve (evolve) over time
  - Or be replaced by a new *Program Of Record*



The Meta-Architecture

- A meta-architecture is a configuration or a pattern into which other architectures fit
- The SoS meta-architecture for this analysis consists of:
  - A list of all the potential component systems (positional), and content (participation)
  - Followed by the first order interfaces of each system with every other system (positional), and content (if this interface exists or is exploited)

Systems																					
S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22
0	1	0	0	0	0	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1	1

Interfaces to Sys 1																				
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
0	0	0	1	1	1	0	1	0	0	0	0	1	1	1	0	1	1	1	1	0

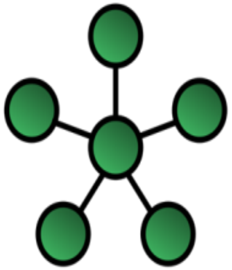
Interfaces to Sys 2																				Interfaces to Sys 3			
i2-3	i2-4	i2-5	i2-6	i2-7	i2-8	i2-9	i2-10	i2-11	i2-12	i2-13	i2-14	i2-15	i2-16	i2-17	i2-18	i2-19	i2-20	i2-21	i2-22	i3-4	i3-5	i3-6	i3-7
1	1	0	0	1	0	1	0	1	0	0	1	1	1	0	1	0	0	1	0	1	0	1	0

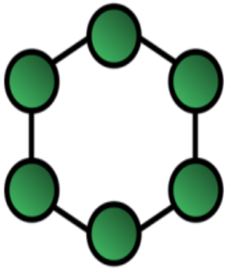
• • •

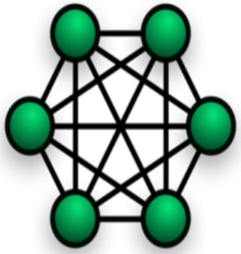
• • •

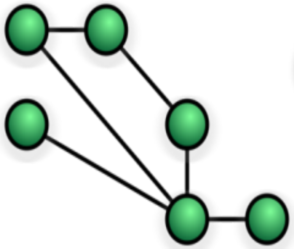
Interfaces to Sys 18				Interfaces to Sys 19			Interfaces to Sys 20		Interfaces to Sys 21	
i18-19	i18-20	i18-21	i18-22	i19-20	i19-21	i19-22	i20-21	i20-22	i21-22	
0	0	1	1	0	1	0	0	1	0	

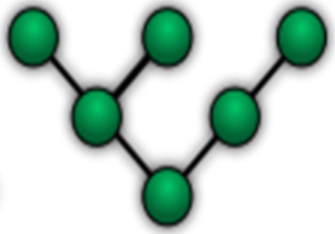
Meta-Architectures for Network Topologies

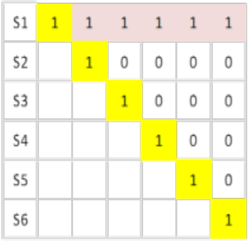




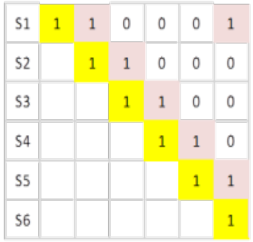




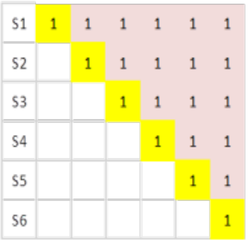




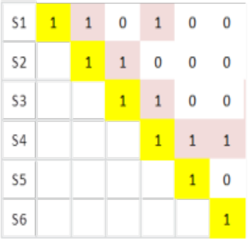
S1	1	1	1	1	1	1
S2		1	0	0	0	0
S3			1	0	0	0
S4				1	0	0
S5					1	0
S6						1



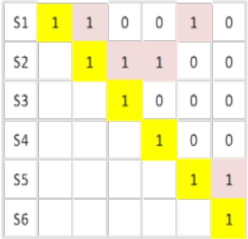
S1	1	1	0	0	0	1
S2		1	1	0	0	0
S3			1	1	0	0
S4				1	1	0
S5					1	1
S6						1



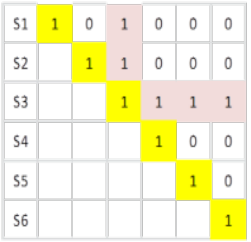
S1	1	1	1	1	1	1
S2		1	1	1	1	1
S3			1	1	1	1
S4				1	1	1
S5					1	1
S6						1



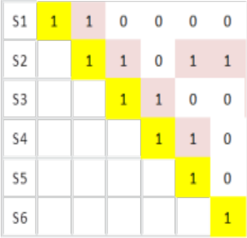
S1	1	1	0	1	0	0
S2		1	1	0	0	0
S3			1	1	0	0
S4				1	1	1
S5					1	0
S6						1



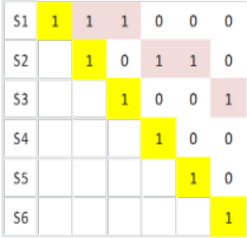
S1	1	1	0	0	1	0
S2		1	1	1	0	0
S3			1	0	0	0
S4				1	0	0
S5					1	1
S6						1



S1	1	0	1	0	0	0
S2		1	1	0	0	0
S3			1	1	1	1
S4				1	0	0
S5					1	0
S6						1



S1	1	1	0	0	0	0
S2		1	1	0	1	1
S3			1	1	0	0
S4				1	1	0
S5					1	0
S6						1



S1	1	1	1	0	0	0
S2		1	0	1	1	0
S3			1	0	0	1
S4				1	0	0
S5					1	0
S6						1

Compact Meta-Architecture

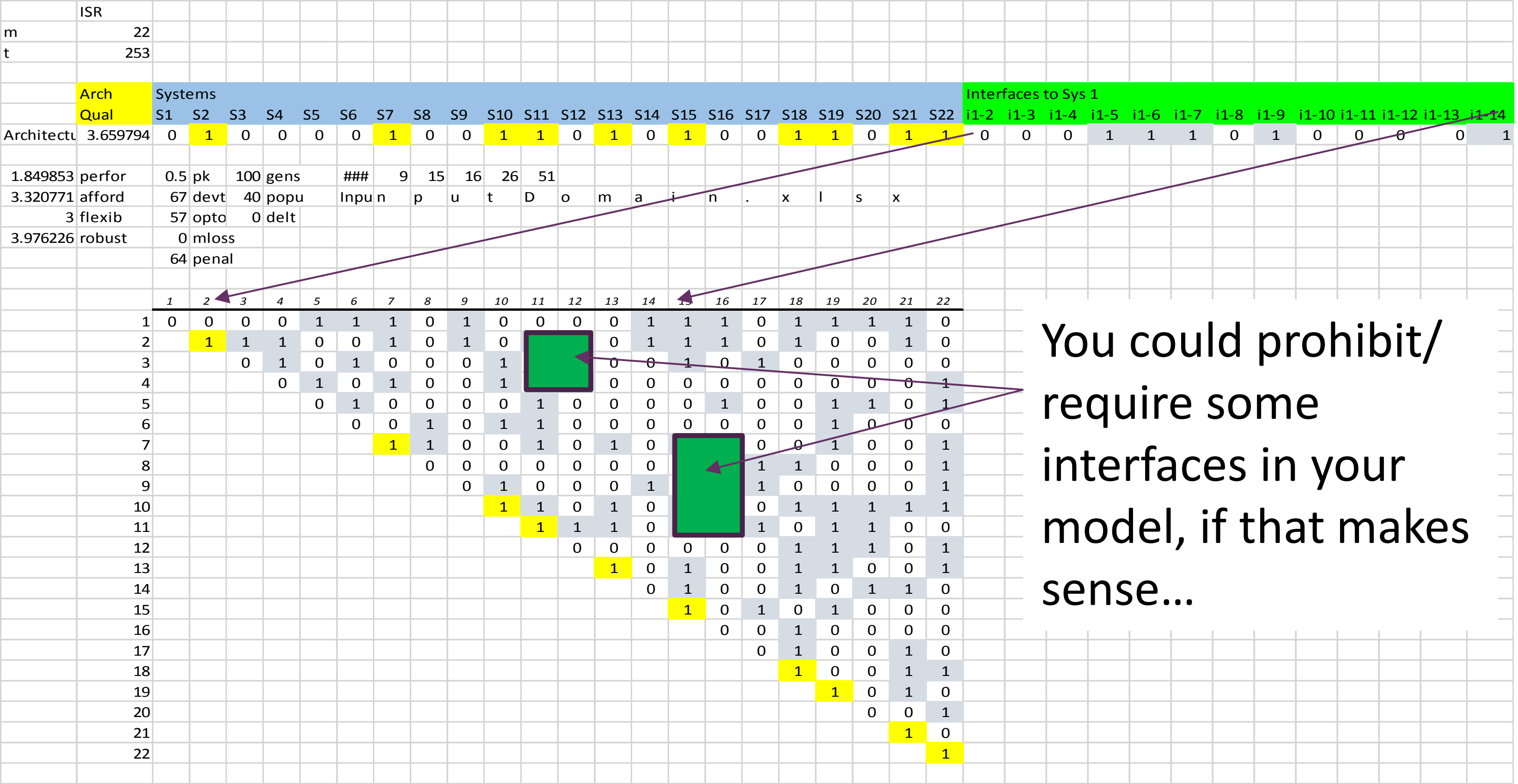
- The binary string is a chromosome for the Genetic Algorithm

- Upper triangular matrix form of the chromosome

X_1	X_{12}	X_{13}	…	X_{1i}	X_{1j}	…	X_{1m}
	X_2	X_{23}	…	X_{2i}	X_{2j}	…	X_{2m}
		X_3	…	X_{3i}	X_{3j}	…	…
			…	…	…	…	…
				X_i	X_{ij}	…	X_{im}
					X_j	…	X_{jm}
						…	$X_{(m-1)m}$
							X_m

- $X_{ij \text{ for } i=j}$ = X_i, the system i
- $X_{ij \text{ for } i<>j}$ = the interface between system i and system j

Participating Systems & Interfaces Highlighted in Architecture Display



	ISR																						
m	22																						
t	253																						
	Arch	Systems																					
	Qual	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22
Architectu	3.659794	0	1	0	0	0	0	1	0	0	1	1	0	1	0	1	0	0	1	1	0	1	1
1.849853	perfor	0.5	pk	100	gens		###	9	15	16	26	51											
3.320771	afford	67	devt	40	popu		Inpu	n	p	u	t	D	o	m	a	i	n	.	x	l	s	x	
3	flexib	57	opto	0	delt																		
3.976226	robust	0	mloss																				
		64	penal																				

Interfaces to Sys 1												
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
0	0	0	1	1	1	0	1	0	0	0	0	1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0	0	0	0	1	1	1	0	1	0	0	0	0	1	1	1	0	1	1	1	1	0
2		1	1	1	0	0	1	0	1	0			0	1	1	1	0	1	0	0	1	0
3			0	1	0	1	0	0	0	1			0	0	1	0	1	0	0	0	0	0
4				0	1	0	1	0	0	1			0	0	0	0	0	0	0	0	0	1
5					0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	1	0	1
6						0	0	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0
7							1	1	0	0	1	0	1	0			0	0	1	0	0	1
8								0	0	0	0	0	0	0			1	1	0	0	0	1
9									0	1	0	0	0	1			1	0	0	0	0	1
10										1	1	0	1	0			0	1	1	1	1	1
11											1	1	1	0			1	0	1	1	0	0
12												0	0	0	0	0	0	1	1	1	0	1
13													1	0	1	0	0	1	1	0	0	1
14														0	1	0	0	1	0	1	1	0
15															1	0	1	0	1	0	0	0
16																0	0	1	0	0	0	0
17																	0	1	0	0	1	0
18																		1	0	0	1	1
19																			1	0	1	0
20																				0	0	1
21																					1	0
22																						1

You could prohibit/ require some interfaces in your model, if that makes sense…

Model Inputs (Regardless of Domain)

- What are you trying to do with the SoS?
  - WRITE IT DOWN…and SHARE IT!
- What existing systems, with what existing capabilities, are available to contribute?
- How could the systems be combined in a way better than is done now?
  - Possibly with limited changes…
- What attributes are important to the stakeholders?
  - How do the stakeholders define them?
- How do the stakeholders value performance in each attribute?
- Estimated cost, schedule, capabilities & performance to join the SoS

Collecting Domain Data

Overarching Purpose of SoS:	ISR & Targeting of Gulf War Scud Transporter/Erector/Launchers (TELs)
Unique value of SoS	Existing non-networked systems not doing job – this might!
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 Affordability Robustness Flexibility
Capabilities of contributing systems	EO/IR Synthetic Aperture Radar Exploitation Command & Control Communications

Typical SoS Attributes

- Performance: How this depends on systems and interfaces below
- Flexibility: Depends on number of systems as sources of required capabilities
- Robustness: Depends on distribution of capabilities across systems
- Affordability: Depends on which systems bring high costs
- Availability: Depends on systems & interfaces reliability
- Agility: Ability to rapidly switch to other missions or infrastructure
- Resilience: Ability to withstand intentional attack or natural disaster

Attribute Values Defined in This Context

- **Performance**: generally, the sum of the performance in required capabilities of the individual component systems, with a small boost (delta) in performance due to increased coordination through interfaces (see next chart)

- **Affordability**: roughly the inverse of the sum of the development and operation costs of the SoS. The performance delta above is applied in a different way to the affordability to change its shape as a function of the number of interfaces

- **Developmental Flexibility**: roughly the inverse of the number of sources that the SoS manager has for each capability. If a required capability is available from only one component system, then the SoS manager s flexibility is very small; they must have that system. On the other hand, if the capability is available from multiple systems within the SoS, the manager has 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 your SoS is a good thing; however, if that system is ever absent, the performance of the SoS is degraded substantially. Therefore, it may be useful to have the contributions of the systems more widely dispersed, than concentrated in one or two high capability systems

NetCentric SoS Performance Improvement

- There is an assumed performance increase from being a SoS
  - —If not, why bother with a SoS – just send more systems
- The performance boost comes from interfacing the systems
  - —Assume the form of this boost is something like this

$$P_{SoS} = \sum P_{Systems} * \ (1 + delta)^{\sum Interfaces}$$

  - —P is performance in a capability (or other attribute driven by interfaces)
  - —Delta is a small percentage, depending on context
  - —Interfaces are between the component systems in the SoS

Input Domain System & Capability Data

SysNo	Type	Capability	I/FDevCost	OpsCost/h	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	fighter	2	0.7	10	15	1		x			x
11	fighter	2	0.7	10	15	1		x			x
12	fighter	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

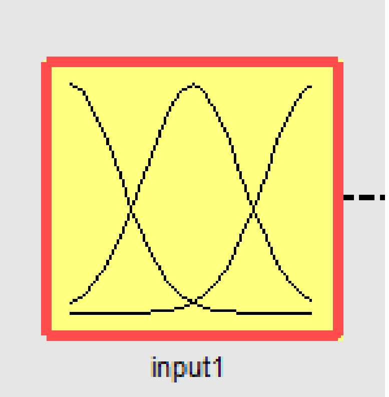
Two More Pieces of the Puzzle

- Attributes are those features/characteristics that come up repeatedly in discussions with stakeholders
  - —Linguistic clustering analysis can help find these
  - —Facilitator pursuit and massage of the data - elicitation
- Membership functions capture how the stakeholders feel about individual attributes
  - —Granularity names of membership functions map to ranges of values
  - —Fuzzy values are mapped to physical world values

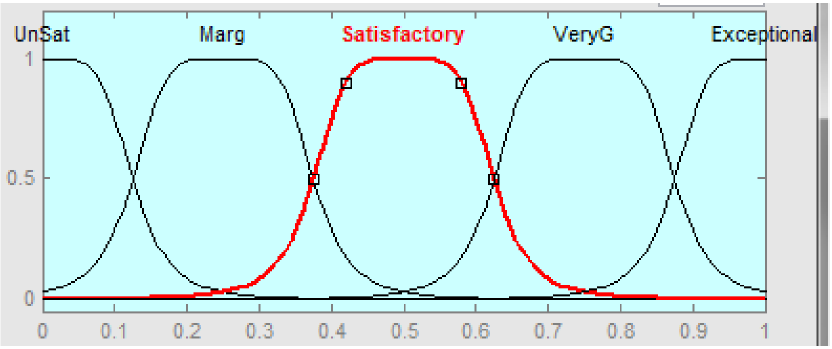
- Rules for combining attribute evaluations to the overall SoS score

Granularity of Evaluations

- Generic evaluation distributions
  - —Unacceptable: At the low end of performance
  - —Middle of the road: Nothing special, average, sort of acceptable - depending on everything else
  - —Very good: High end of performance spectrum



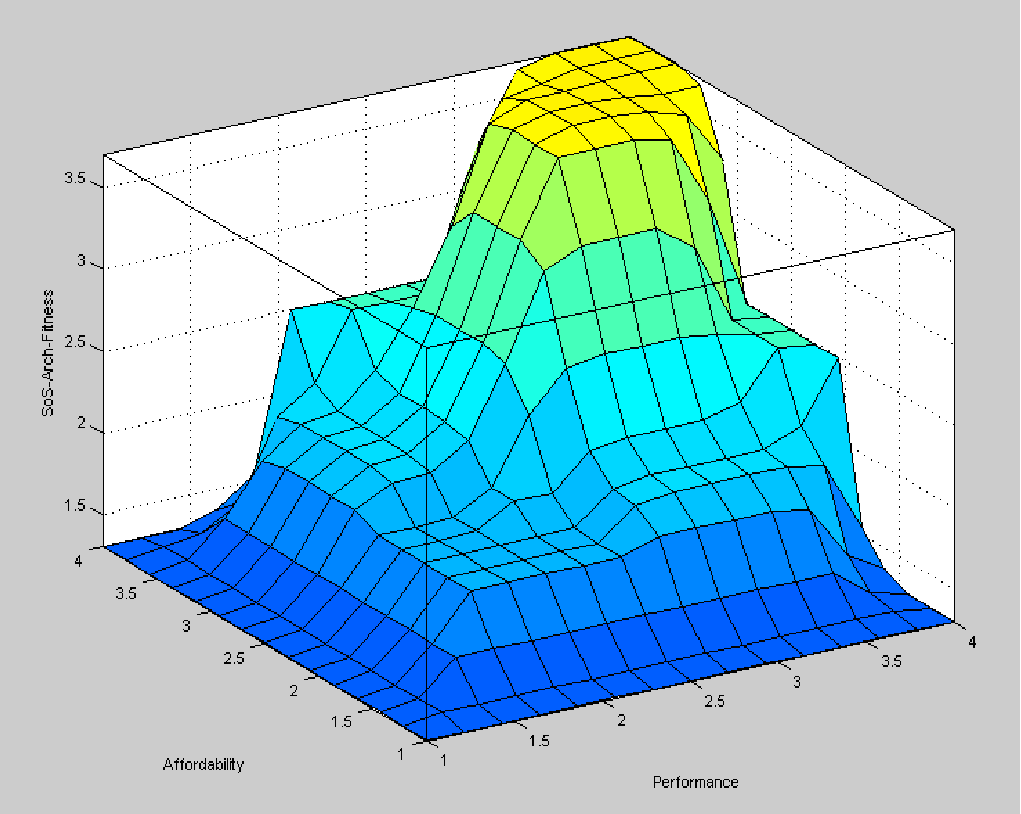
- Gov't Cost Performance Assessment Reports (CPARs)
  - —5 Levels: Exceptional, Very Good, Satisfactory, Marginal, Unsatisfactory
  - — You may not have noticed that these are fuzzy categories!



- Even granularity forces a choice

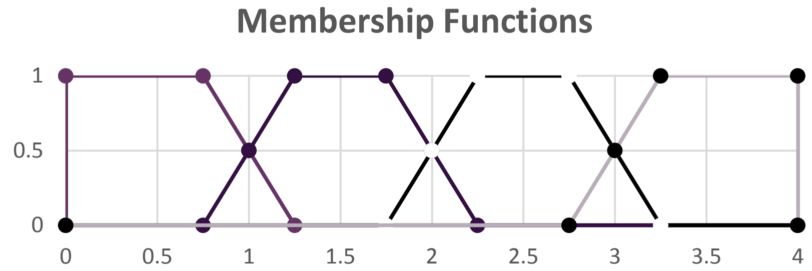
The Rules

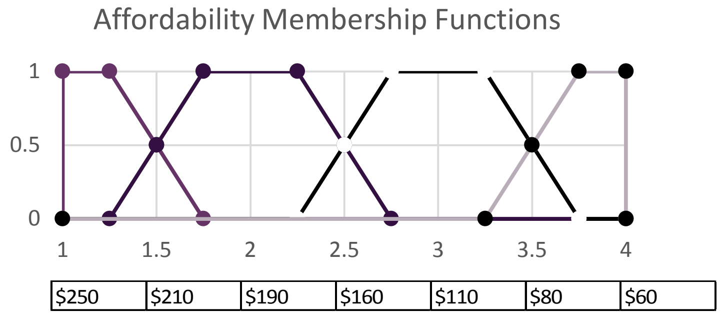
Plain Language Rule
If ANY attribute is Unacceptable, then SoS is Unacceptable
If ALL the attributes are Marginal, then the SoS is Unacceptable
If ALL the attributes are Acceptable, then the SoS is Exceeds
If (Performance AND Affordability) are Exceeds, but (Dev. Flexibility and Robustness) are Marginal,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



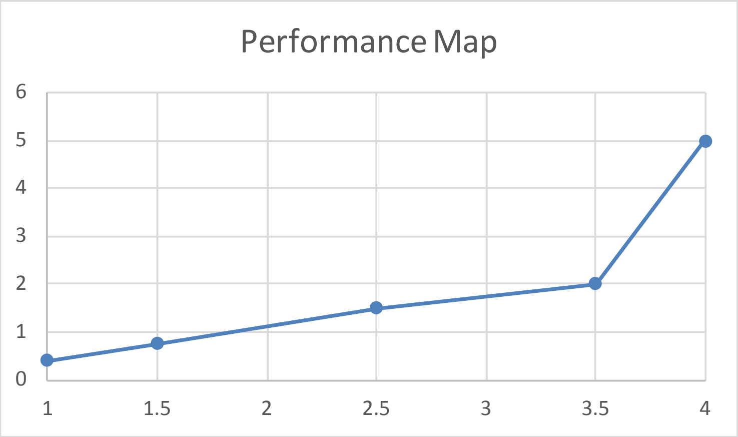
Fuzzy Space and Normal Space

- The range of the fuzzy space is the ‘universe of discourse’
- The membership functions (MF) *span* the universe of discourse
  - —MF may overlap to varying degrees, and have partial values
  - —The fuzzy numbers are relatively unimportant – there only for keeping track of relatively better or worse positions in the space
  - —‘Normal space’ *maps* to ‘fuzzy space’ and vice versa



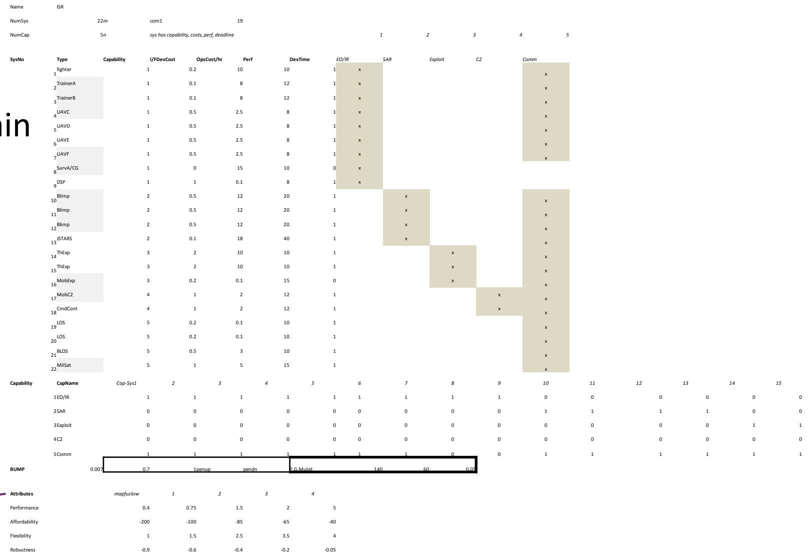


Membership Function Definitions



Attributes	*1*	*1.5*	*2.5*	*3.5*	*4*
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

Domain Data



Name	ISR										
NumSys		22m	com1		19						
NumCap		5n	*sys has capability, costs, perf, deadline*				*1*	*2*	*3*	*4*	*5*
SysNo	**Type**	**Capability**	**I/FDevCost**	**OpsCost/hr**	**Perf**	**DevTime**	*EO/IR*	*SAR*	*Exploit*	*C2*	*Comm*
1	fighter	1	0.2	10	10	1	x				x
2	TrainerA	1	0.1	8	12	1	x				x
3	TrainerB	1	0.1	8	12	1	x				x
4	UAVC	1	0.5	2.5	8	1	x				x
5	UAVD	1	0.5	2.5	8	1	x				x
6	UAVE	1	0.5	2.5	8	1	x				x
7	UAVF	1	0.5	2.5	8	1	x				x
8	SurvA/CG	1	0	15	10	0	x				
9	DSP	1	1	0.1	8	1	x				
10	Blimp	2	0.5	12	20	1		x			x
11	Blimp	2	0.5	12	20	1		x			x
12	Blimp	2	0.5	12	20	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	MobExp	3	0.2	0.1	15	0			x		x
17	MobC2	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	MilSat	5	1	5	15	1					x

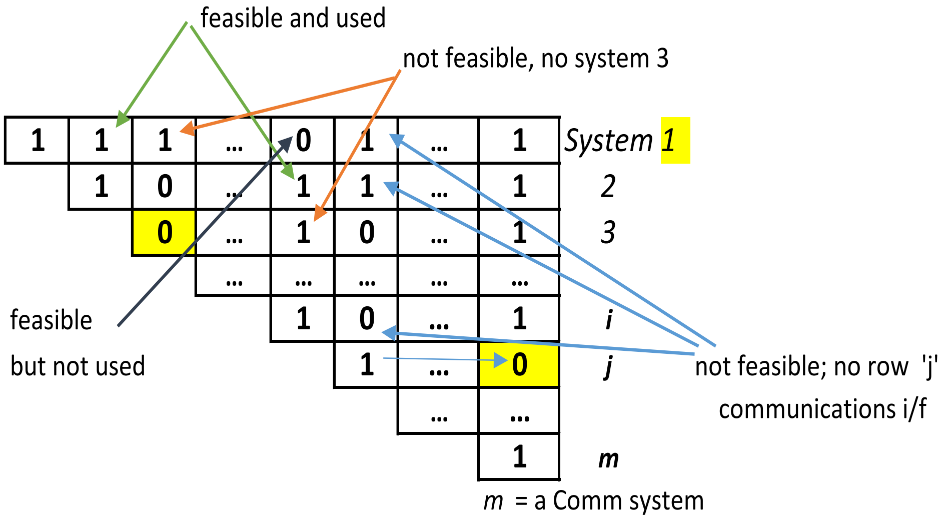
Capability	**CapName**	*Cap-Sys1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
1	EO/IR	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
2	SAR	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
3	Exploit	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
4	C2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	Comm	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
BUMP	0.007	0.7	1	penup	pendn	P.G.Mutat		140	60	0.05						

Attributes	*mapfuzlow*	*1*	*2*	*3*	*4*
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

Fuzzy Membership Function Crossover Points

A Feasibility Twist to Value of Interfaces

- Added the concept of the *feasible interface*
- Two systems may *claim* that they have an interface, but unless supported by a communication link, it is not a *feasible interface*
  - —Arises from the GA use of an initial population of random chromosomes
  - —Now, using a *feasible I/F* is good: Rewarded
  - —Using an *infeasible I/F* is bad: Penalized



feasible and used

not feasible, no system 3

1	1	1	…	0	1	…	1	*System 1*
	1	0	…	1	1	…	1	*2*
		0	…	1	0	…	1	*3*
			…	…	…	…	…	
				1	0	…	1	*i*
					1	…	0	*j*
						…	…	
							1	***m***

feasible but not used

not feasible; no row 'j' communications i/f

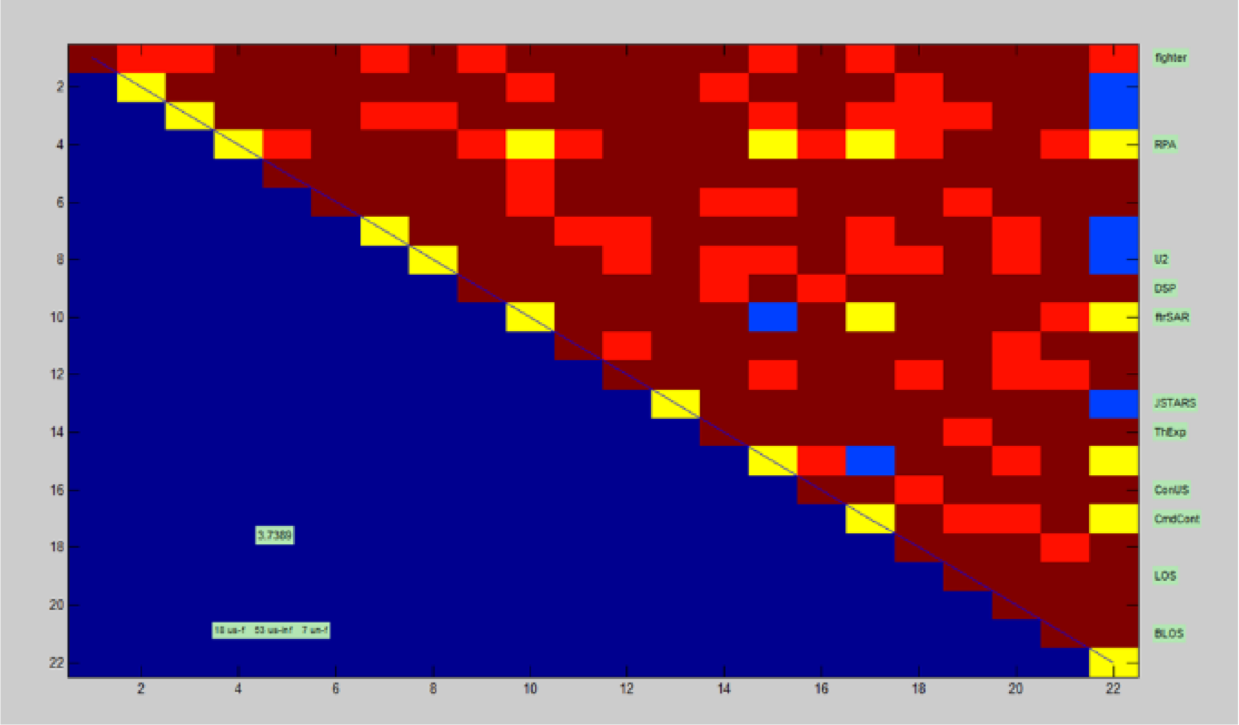
m = a Comm system

Features of the Feasibility Model

- Comm links *enable* the interfaces between systems
  - —Comm link is a communication system with an interface to both systems
- One system's interface can be planned, developed, paid for, installed…but not be useful to the SoS unless the other system and a link are both present
  - —This is a waste of funding & effort
  - —Penalized in SoS performance and cost
- The GA approach needs the penalties and rewards, since it populates and mutates the chromosomes randomly
- Changing one bit usually doesn't change the performance much
  - —Unless it's in the comm systems and their interfaces

Feasible Interfaces – ISR Model

22 Systems in ISR model

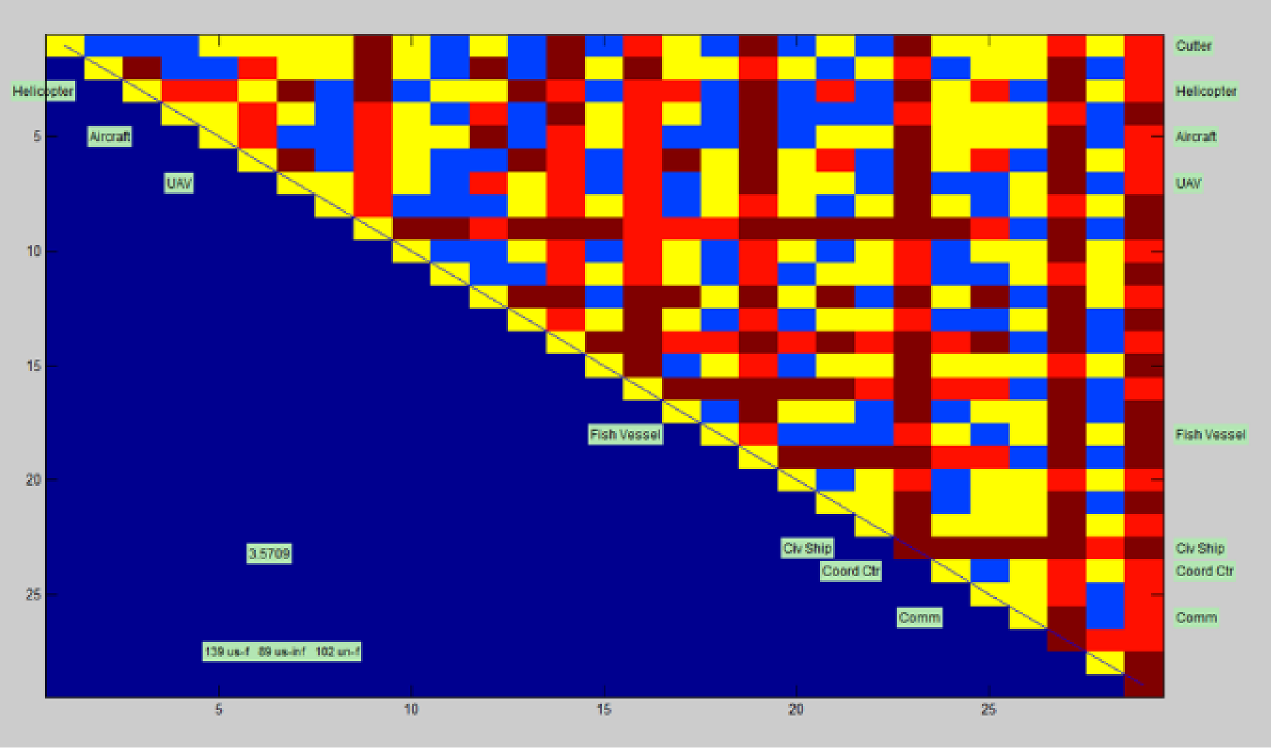


	Feasible	Infeasible
Used	1	1
Not Used	0	0

22 Systems
253 Systems + Interfaces
18 Used – Feasible
53 Used – Infeasible
7 Not used - Feasible

Search & Rescue Scenario Solution Example

29 Systems in SAR model



	Feasible	Infeasible
Used	1	1
Not Used	0	0

29 Systems

435 Systems + Interfaces

139 Used – Feasible

59 Used – Infeasible

102 Not used – Feasible

Feasibility Concept Impact

- Now the sum of the interfaces become sum of good interfaces minus the sum of bad (infeasible) interfaces
- $P_{SoS} = \sum P_{Systems} * \ (1 + delta)^{(\sum Feas.Interfaces - \sum Infeas.Interfaces)}$
- 'Hedging your NCO bets' by spending to develop lots of interfaces for potential use is not a good idea
  - —Tunable parameters: delta, Feas, Infeas
  - —Exponent can go negative
- On the other hand, interfaces don't cost money to operate – systems cost money to operate (life cycle consideration)

Model Process Summary To This Point

- SoS purpose defined
- Domain data for systems, interfaces & changes estimated
- Desirable attributes defined with measures
  - Measures depend on choice of systems and their interconnections
- Fuzzy variables tentatively mapped to measures
- Rules for combining fuzzy attributes to an overall SoS measure
- Ready to let the Genetic Algorithm attack the problem of ‘what should we pick’ to design our SoS

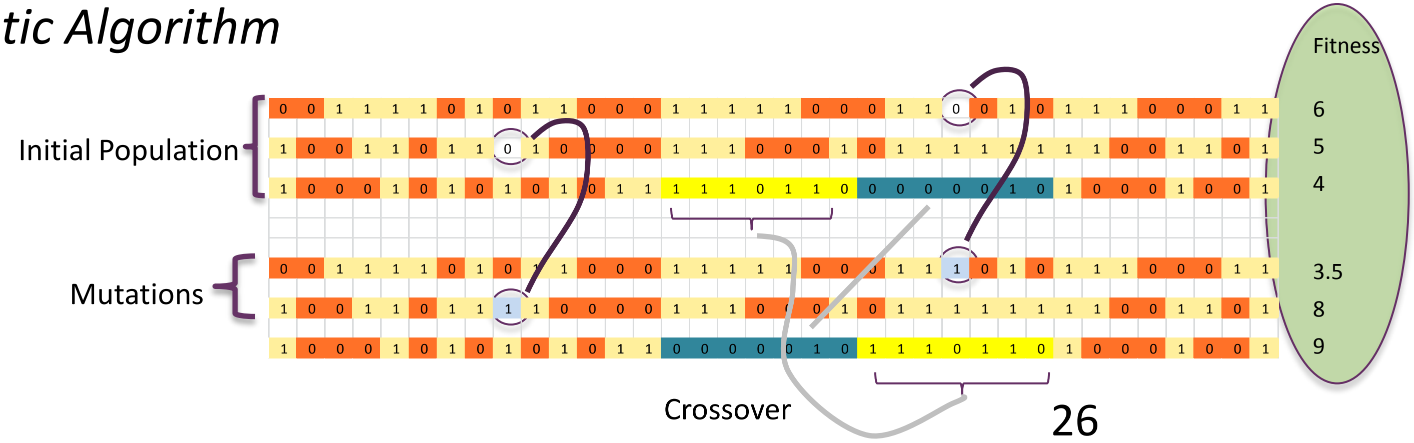
Genetic Algorithm

Chromosome representation – first Systems, then Interfaces

s_1	s_2		s_i		s_n	s_{12}		s_{1j}		s_{1n}	s_{23}			$s_{n-1,n}$

$$s_{ij} = s_{ji} = \begin{cases} 1, & \text{if exists an edge between vertices } i \text{ and } j \\ 0, & \text{otherwise} \end{cases}$$

Participation in the SoS, and existence of interfaces between the systems, is ideally suited for a *chromosome representation in a Genetic Algorithm*

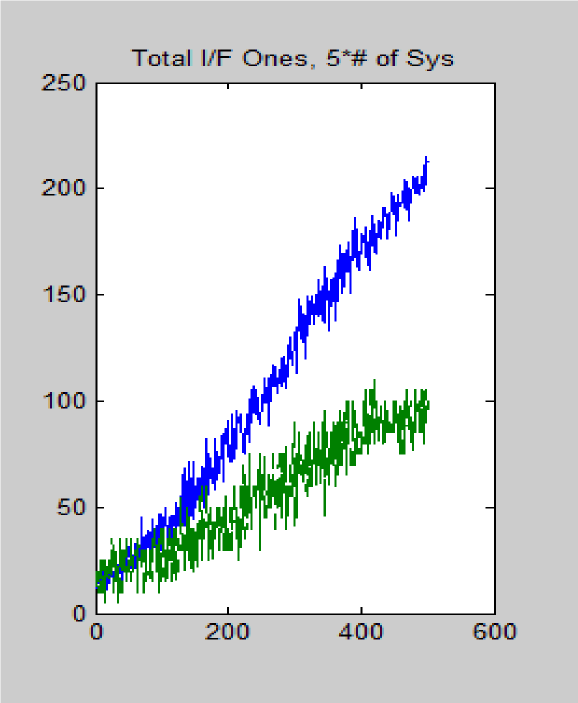


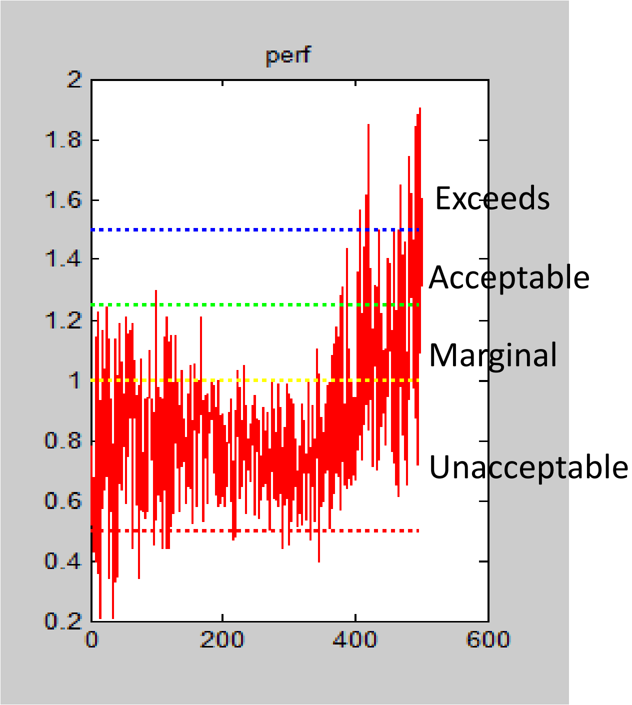
Initial Population	0	0	1	1	1	1	0	1	0	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	0	0	1	0	1	1	1	0	0	0	1	1	6
	1	0	0	1	1	0	1	1	0	1	0	0	0	0	1	1	1	0	0	0	1	0	1	1	1	1	1	1	1	1	0	0	1	1	0	1	5
	1	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	0	1	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	1	4
Mutations	0	0	1	1	1	1	0	1	0	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1	0	1	0	1	1	1	0	0	0	1	1	3.5
	1	0	0	1	1	0	1	1	1	1	0	0	0	0	1	1	1	0	0	0	1	0	1	1	1	1	1	1	1	1	0	0	1	1	0	1	8
	1	0	0	0	1	0	1	0	1	0	1	0	1	1	0	0	0	0	0	1	0	1	1	1	0	1	1	0	1	0	0	0	1	0	0	1	9

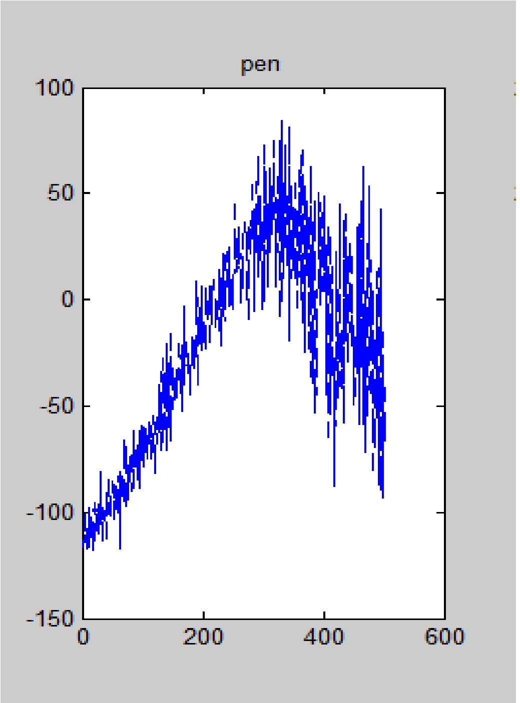
Combining Fuzzy and GA Approach

- Sample a few hundred random chromosomes
  - Independent variable is how many 1’s in the architecture
  - Plot attribute and SoS evaluations
- Perform the end-around check to see that it all hangs together
  - Insure that you get *some* good SoS chromosomes
- One more trick: for follow on waves, allow the keeping of selected systems/interfaces in your architecture
  - An ‘input chromosome’ to protect the last wave’s negotiated systems and interfaces from being mutated away
  - Input chromosome is all zeroes for the first wave

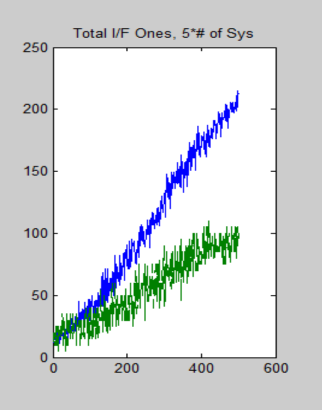
Exploring the Meta-Architecture

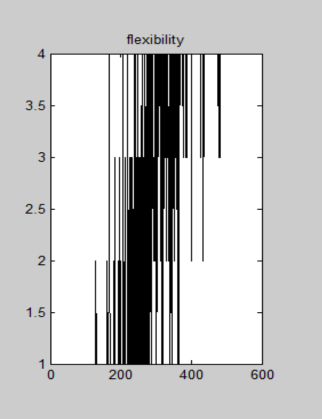


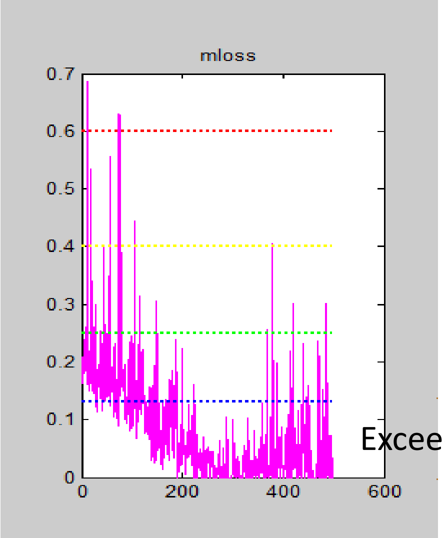




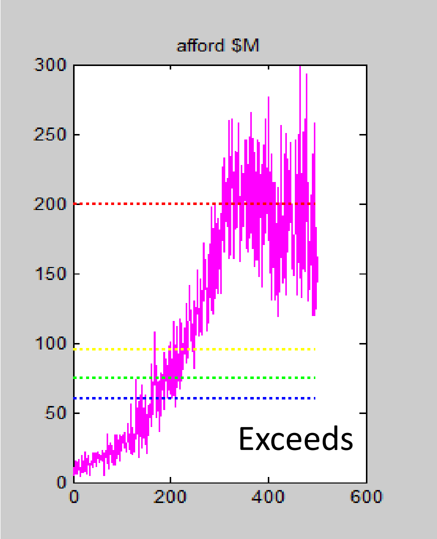
Exploring the Meta-Architecture



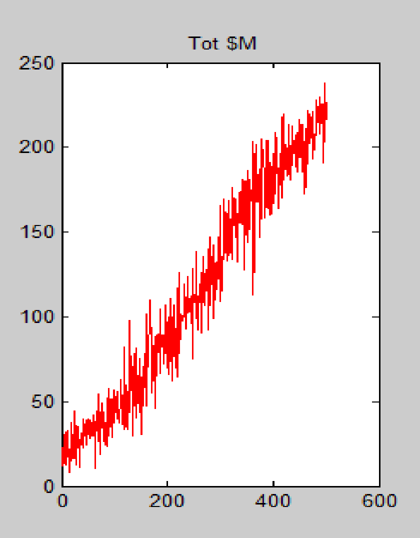




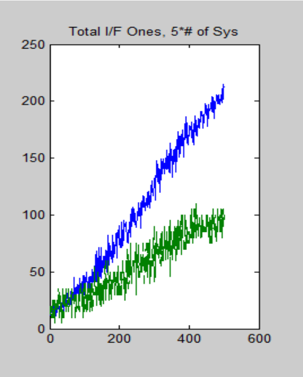
Exceeds

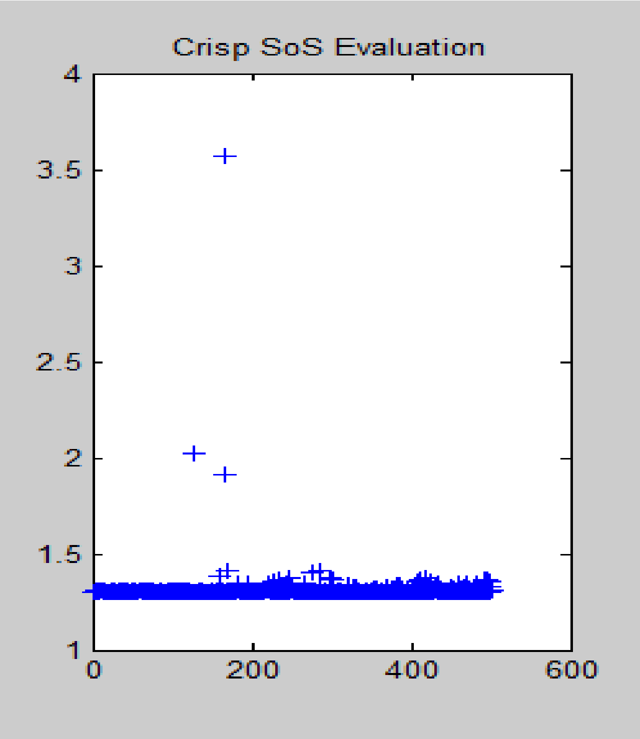


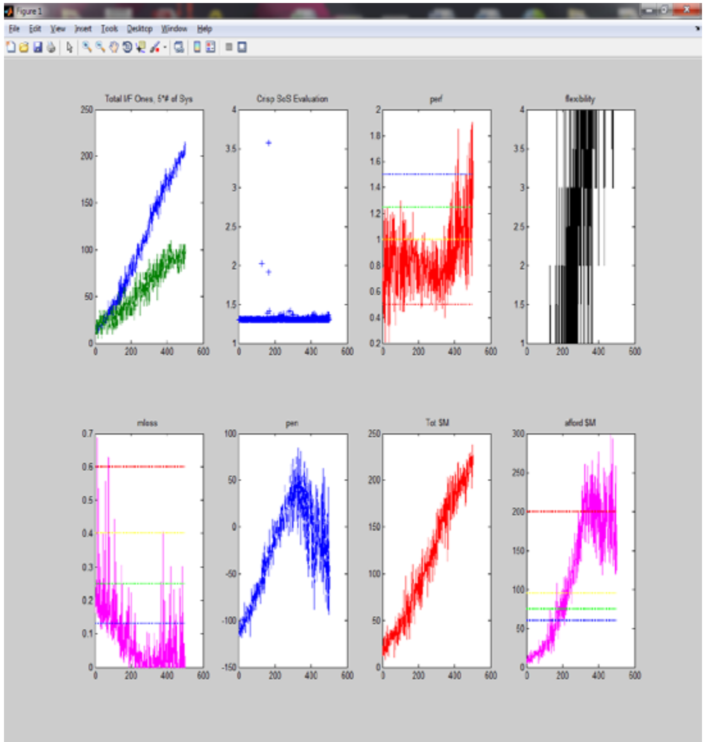
Exceeds



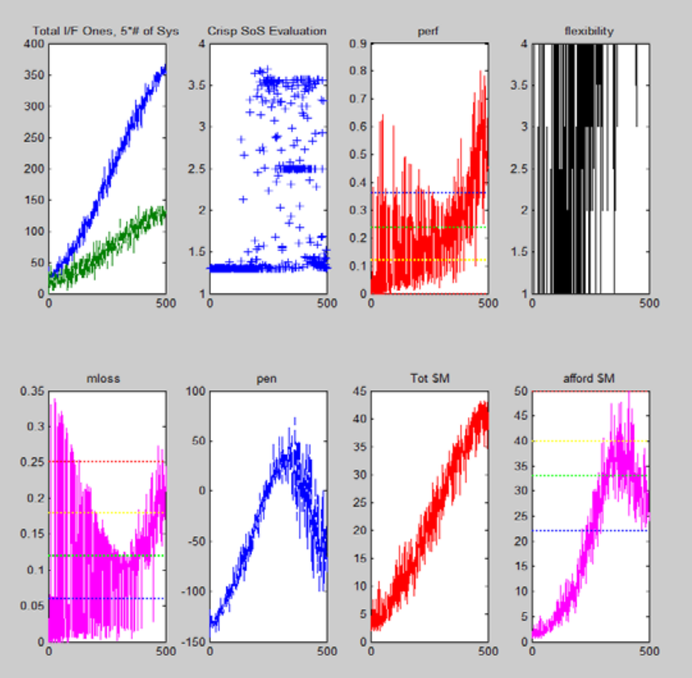
Exploring the Meta-Architecture

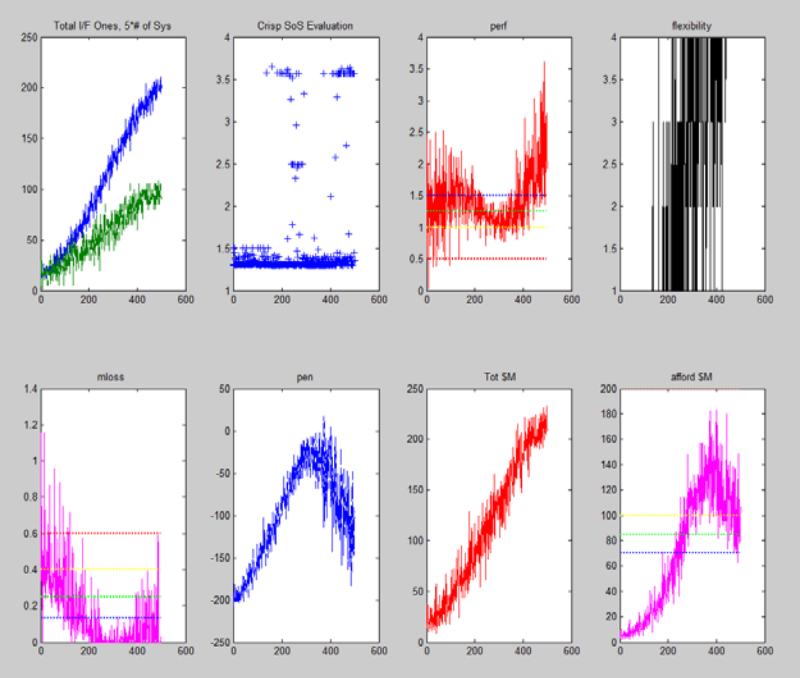






Exploring Other Meta-Architectures





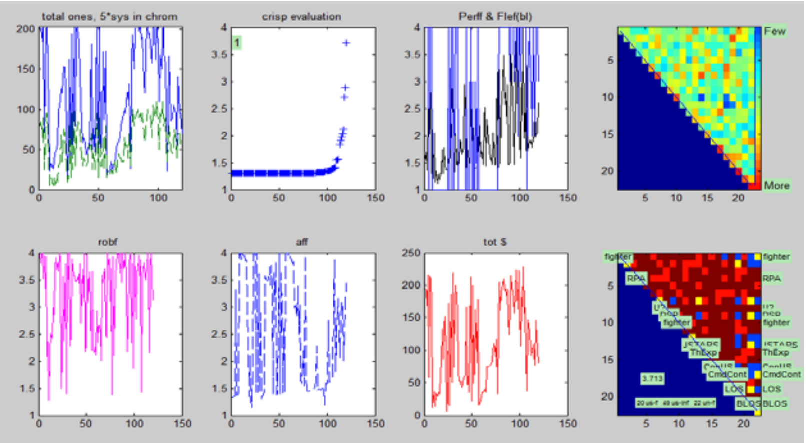
BUMP	0.0035	0.5	0.2	penup	pendn	P,G,Mutat
Attributes		mapfuzlow	1.5	2.5	3.5	4
Performance		0	0.15	0.24	,4	0.55
Affordability		-60	-45	-35	-24	-10
Flexibility		0	1	2	3	4
Robustness		-0.4	-0.2	-0.15	-0.08	0

BUMP	0.007	0.7	1	penup	pendn	P,G,Mutat
Attributes		mapfuzlow	1.5	2.5	3.5	4
Performance		0.5	1	1.25	1.5	5
Affordability		-200	-100	-85	-70	-40
Flexibility		0	1	2	3	4
Robustness		-0.6	-0.4	-0.25	-0.13	-0.05

Genetic Algorithm Generations

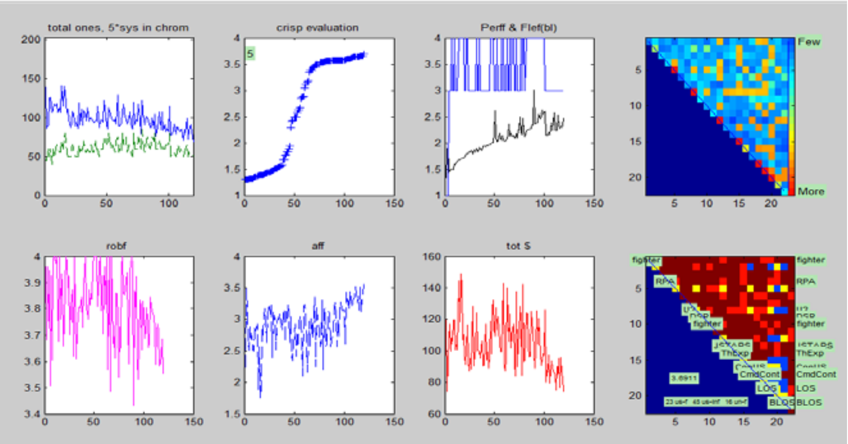
Gen 1

3.69



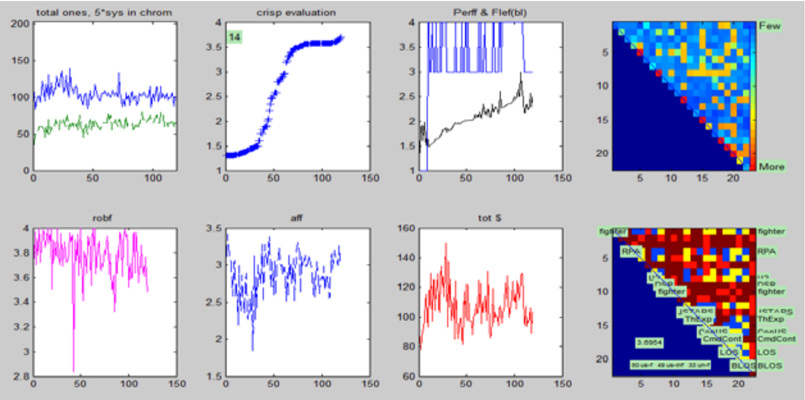
Gen 6

3.691



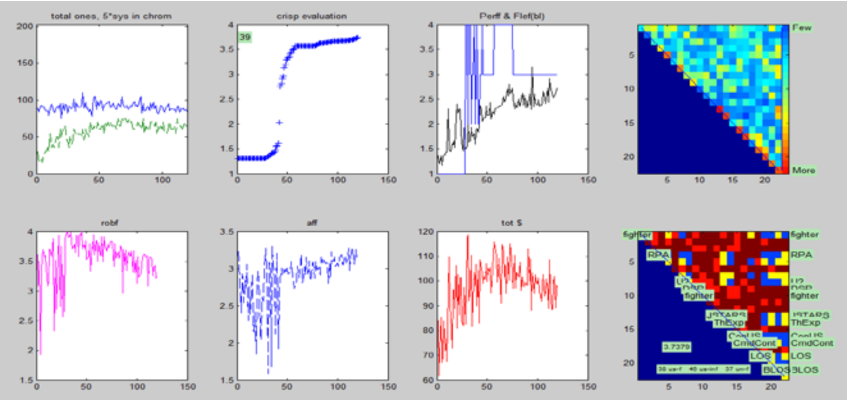
Gen 14

3.695

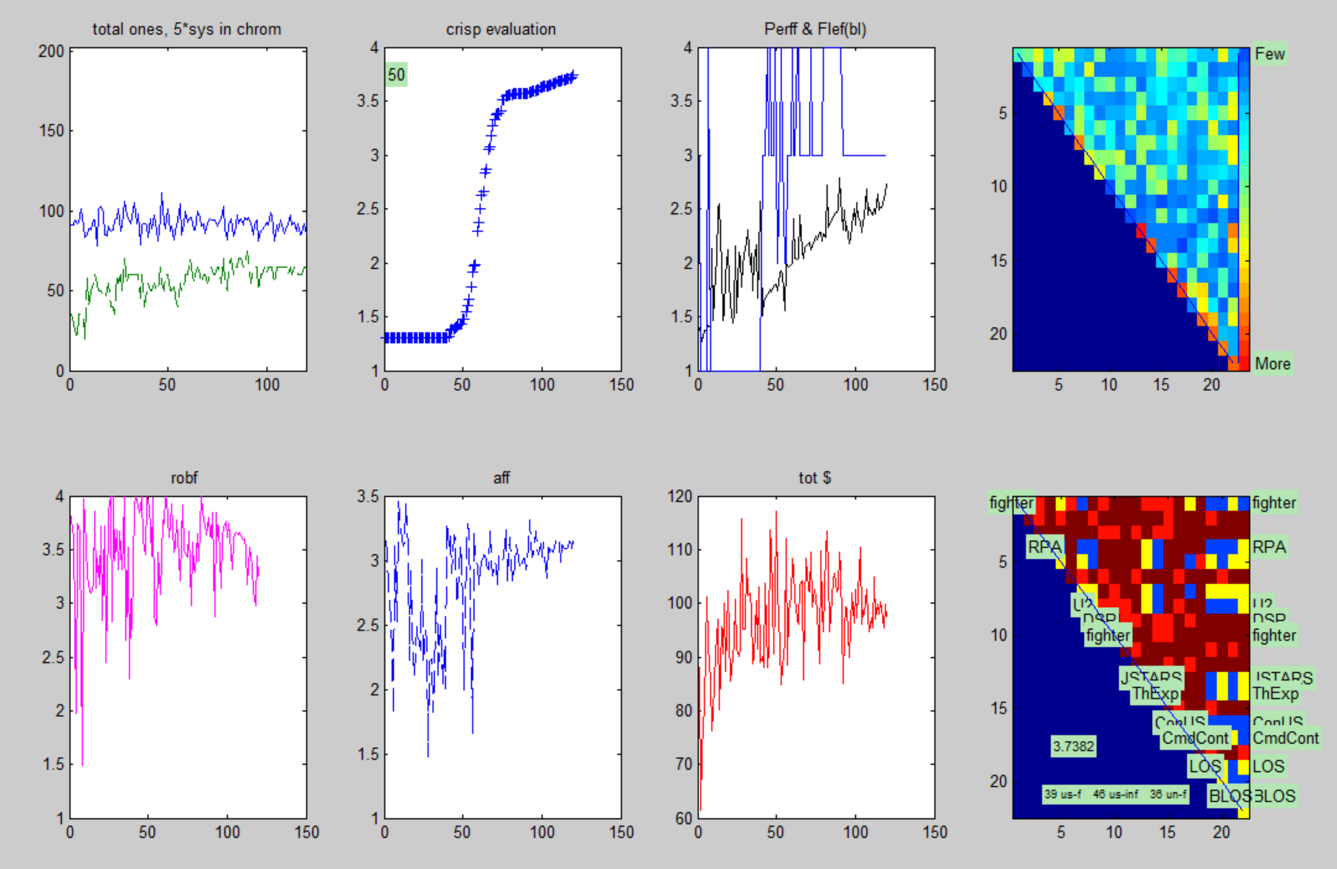


Gen 39

3.738



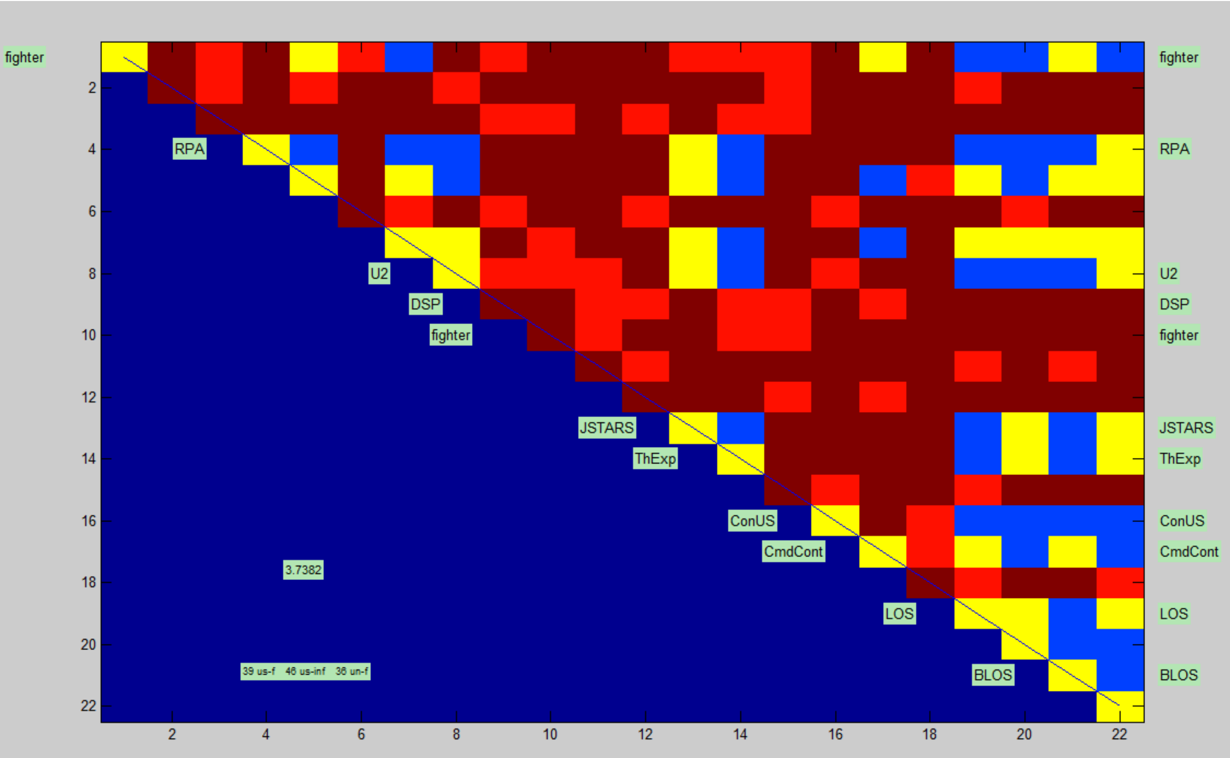
50 Generations In

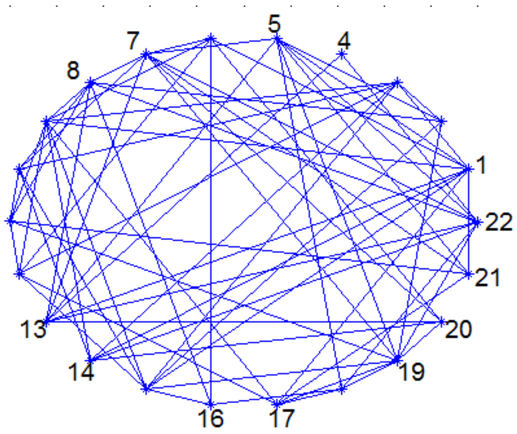


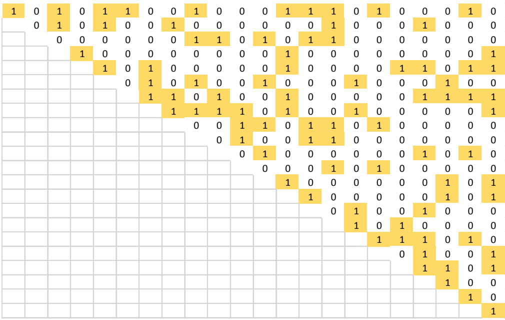
Gen 50

3.7382

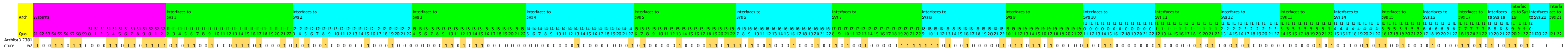
Suggested Architecture







1	0	1	0	1	1	0	0	1	0	0	0	1	1	1	0	1	0	0	0	1	0
	0	1	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0
		0	0	0	0	0	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0
			1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
				1	0	1	0	0	0	0	0	1	0	0	0	0	1	1	0	1	1
					0	1	0	1	0	0	1	0	0	0	1	0	0	0	1	0	0
						1	1	0	1	0	0	1	0	0	0	0	0	1	1	1	1
							1	1	1	1	0	1	0	0	1	0	0	0	0	0	1
								0	0	1	1	0	1	1	0	1	0	0	0	0	0
									0	1	0	0	1	1	0	0	0	0	0	0	0
										0	1	0	0	0	0	0	0	1	0	1	0
											0	0	0	1	0	1	0	0	0	0	0
												1	0	0	0	0	0	0	1	0	1
													1	0	0	0	0	0	1	0	1
														0	1	0	0	1	0	0	0
															1	0	1	0	0	0	0
																1	1	1	0	1	0
																	0	1	0	0	1
																		1	1	0	1
																			1	0	0
																				1	0
																					1



	Arch	Systems																					
	Qual	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22
Architecture	3.738167	1	0	0	1	1	0	1	1	0	0	0	0	1	1	0	1	1	0	1	1	1	1

	Interfaces to Sys 1																				
	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
Architecture	0	1	0	1	1	0	0	1	0	0	0	1	1	1	0	1	0	0	0	1	0

	Interfaces to Sys 2																			
	i2-3	i2-4	i2-5	i2-6	i2-7	i2-8	i2-9	i2-10	i2-11	i2-12	i2-13	i2-14	i2-15	i2-16	i2-17	i2-18	i2-19	i2-20	i2-21	i2-22
Architecture	1	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0

	Interfaces to Sys 3																		
	i3-4	i3-5	i3-6	i3-7	i3-8	i3-9	i3-10	i3-11	i3-12	i3-13	i3-14	i3-15	i3-16	i3-17	i3-18	i3-19	i3-20	i3-21	i3-22
Architecture	0	0	0	0	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0

	Interfaces to Sys 4																	
	i4-5	i4-6	i4-7	i4-8	i4-9	i4-10	i4-11	i4-12	i4-13	i4-14	i4-15	i4-16	i4-17	i4-18	i4-19	i4-20	i4-21	i4-22
Architecture	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1

	Interfaces to Sys 5																
	i5-6	i5-7	i5-8	i5-9	i5-10	i5-11	i5-12	i5-13	i5-14	i5-15	i5-16	i5-17	i5-18	i5-19	i5-20	i5-21	i5-22
Architecture	0	1	0	0	0	0	0	1	0	0	0	0	1	1	0	1	1

	Interfaces to Sys 6															
	i6-7	i6-8	i6-9	i6-10	i6-11	i6-12	i6-13	i6-14	i6-15	i6-16	i6-17	i6-18	i6-19	i6-20	i6-21	i6-22
Architecture	1	0	1	0	0	1	0	0	0	1	0	0	0	1	0	0

	Interfaces to Sys 7														
	i7-8	i7-9	i7-10	i7-11	i7-12	i7-13	i7-14	i7-15	i7-16	i7-17	i7-18	i7-19	i7-20	i7-21	i7-22
Architecture	1	0	1	0	0	1	0	0	0	0	0	1	1	1	1

	Interfaces to Sys 8													
	i8-9	i8-10	i8-11	i8-12	i8-13	i8-14	i8-15	i8-16	i8-17	i8-18	i8-19	i8-20	i8-21	i8-22
Architecture	1	1	1	0	1	0	0	1	0	0	0	0	0	1

	Interfaces to Sys 9												
	i9-10	i9-11	i9-12	i9-13	i9-14	i9-15	i9-16	i9-17	i9-18	i9-19	i9-20	i9-21	i9-22
Architecture	0	1	1	0	1	1	0	1	0	0	0	0	0

	Interfaces to Sys 10											
	i10-11	i10-12	i10-13	i10-14	i10-15	i10-16	i10-17	i10-18	i10-19	i10-20	i10-21	i10-22
Architecture	1	0	0	1	1	0	0	0	0	0	0	0

	Interfaces to Sys 11										
	i11-12	i11-13	i11-14	i11-15	i11-16	i11-17	i11-18	i11-19	i11-20	i11-21	i11-22
Architecture	1	0	0	0	0	0	0	1	0	1	0

	Interfaces to Sys 12									
	i12-13	i12-14	i12-15	i12-16	i12-17	i12-18	i12-19	i12-20	i12-21	i12-22
Architecture	0	0	1	0	1	0	0	0	0	0

	Interfaces to Sys 13								
	i13-14	i13-15	i13-16	i13-17	i13-18	i13-19	i13-20	i13-21	i13-22
Architecture	0	0	0	0	0	0	1	0	1

	Interfaces to Sys 14							
	i14-15	i14-16	i14-17	i14-18	i14-19	i14-20	i14-21	i14-22
Architecture	0	0	0	0	0	1	0	1

	Interfaces to Sys 15						
	i15-16	i15-17	i15-18	i15-19	i15-20	i15-21	i15-22
Architecture	1	0	0	1	0	0	0

	Interfaces to Sys 16					
	i16-17	i16-18	i16-19	i16-20	i16-21	i16-22
Architecture	0	1	0	0	0	0

	Interfaces to Sys 17				
	i17-18	i17-19	i17-20	i17-21	i17-22
Architecture	1	1	0	1	0

	Interfaces to Sys 18				Interfaces to Sys 19			Interfaces to Sys 20		Interfaces to Sys 21
	i18-19	i18-20	i18-21	i18-22	i19-20	i19-21	i19-22	i20-21	i20-22	i21-22
Architecture	1	0	0	1	1	0	1	0	0	0

Steps of the Wave Model

- First 2 steps of the FILA-SoS wave are complete
  - —*Initiate* & *Analyze* SoS
- *Develop* step is only half complete
  - —Suggested an architecture, but…
  - —Must now get 'buy in' and agreement from the systems through negotiations
- Agent Based Model (ABM) takes over for the negotiations
  - —Influenced by environment (policies, needs, budgets, etc.)
  - —System negotiator Agents selectable from among Cooperative, Selfish, or Opportunistic models
  - —SoS manager Agent gets the best SoS possible, within constraints
  - —OPM and CPN modeling can be done on suggested or negotiated architecture
- Next wave: New environment, possible new systems (with new data), new analysis, etc.

Summary

- Created and explained a binary SoS Meta-Architecture model
- Discussed how to model an SoS so that its quality depends on the participation of systems and their mutual interfaces
- Showed how the SoS model can be explored with a fuzzy genetic algorithm to analyze the SoS
- Showed that finding 'good' suggested architectures is possible
- Showed how the architecture generation and evaluation can feed the agent based negotiation process to find a 'realizable' SoS
- Used the process in developing several epochs in the Wave Model

Lessons Learned

- No one should have thought SoS were not complicated
  - Many, many little steps in the development of the approach
  - Many little assumptions at different stages
  - Easy to get lost; need configuration control, naming conventions…
- Visualization techniques are invaluable
  - Lots and lots of data – both input and output
  - Data presentation should be considered right from the start
  - Visualize intermediate computation stages, to understand processes
- Modularity with well defined interfaces is necessary

Acknowledgement

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.

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

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

Some References

Agarwal, S,. Pape, L .E., & Dagli, C.H., A Hybrid Genetic Algorithm and Particle Swarm Optimization with Type-2 Fuzzy Sets for Generating Systems of Systems Architectures, proceedings of Complex Adaptive Systems conference, Philadelphia, 2014.

Agarwal, S., Louis E. Pape, Nil Kilicay-Ergin, Cihan H. Dagli, Multi-agent Based Architecture for Acknowledged System of Systems, Procedia Computer Science, Volume 28, 2014, Pages 1-10, ISSN 1877-0509.

Dahmann, J., Rebovich, G., Lowry, R., Lane, J., & Baldwin, K. (2011, April). An implementers' view of systems engineering for systems of systems. In Systems Conference (SysCon), 2011 IEEE International (pp. 212-217). IEEE.

Pape, L. E. , Agarwal, S,. Giammarco, K., & Dagli, C.H., , Fuzzy Optimization of Acknowledged System of Systems Meta-architectures for Agent based Modeling of Development, Procedia Computer Science, Volume 28, 2014, Pages 404-411, ISSN 1877-0509.

Pape, L., & Dagli, C. Assessing robustness in systems of systems meta-architectures. Procedia Computer Science, 20, 262-269, (2013).

Pape L., Giammarco K., Colombi J., Dagli, C.,Kilicay-Ergin N, Rebovich G., “A Fuzzy Evaluation Method for System-of-Systems Meta-architectures” Conference on Systems Engineering Research, Procedia Computer Science, Vol. 16, pp. 245-254, 2013.

Wang, R., Agarwal,S., & Dagli, C., Executable System of Systems Architecture Using OPM in Conjunction with Colored Petri Net: A Module for Flexible Intelligent & Learning Architectures for System of Systems, Europe Middle East & Africa Systems Engineering Conference (EMEASEC), 2014.