SysCon 2008 – IEEE International Systems Conference
Montreal, Canada, April 7–10, 2008

System of Systems Architecture Generation and Evaluation using Evolutionary Algorithms

Joseph J. Simpson[1], Dr. Cihan H. Dagli[2]
[1]Missouri University of Science and Technology, 6400 32nd Ave. NW, #9 Seattle, WA 98107
[2]Missouri University of Science and Technology, 229 Eng. Mgt. & systems Engineering Dept. Rolla, MO 65409

Abstract – *Evolutionary algorithms and computational intelligence represent a developing technology and science that provides great potential in the area of system and system-of-systems architecture generation, categorization and evaluation. Classical system engineering analysis techniques have been used to represent a system architecture in a manner that is compatible with evolutionary algorithms and computational intelligence techniques. This paper focuses on specific system relationship configurations and attributes that are required to successfully aggregate the best-fit function in a fuzzy associative memory that is used in an evolutionary algorithm to generate and evaluate system architectures.*
Keywords *– Evolutionary Algorithms, system, system of systems, computational intelligence.*

I. INTRODUCTION

Evolutionary computation and evolutionary algorithms are a component of computational intelligence that represents a developing science and technology that can be effectively applied to the generation and evaluation of system and system-of-systems architectures. A general technique used by systems engineering professionals is a binary matrix representation of a system or system of systems. The specific meaning and semantics of the binary relationship depends on the type of representation used. Typical representations are "N squared", design structure matrix, dependency structure matrix, and implication matrix. A key feature of these typical representations is their direct relationship to the structure required in an evolutionary computational approach. Evolutionary algorithms can be applied to the evaluation and optimization of these matrix structures. A new evolutionary algorithm has been developed that applies specifically to the generation and evaluation of systems and system of systems. This new evolutionary algorithm incorporates a fuzzy inference system in the calculation of the best fit evaluation. The current industrial and social environment is populated with a vast array of existing and developing systems. Any new system must take this context into account. Formal system analysis has been used to specify each given context and interface as a binary matrix. Evolutionary computation is applied to assist the system architect and engineer in the evaluation of these complex configurations and interface sets.

The rest of this paper is structured as follows: Section II outlines the issues and benefits which motivate the use of computational intelligence and evolutionary algorithms for system and system-of-systems architecture generation and evaluation; Section III describes the general evolutionary algorithm approach using a general example; Section IV is focused on the exploration and explanation of the technique used to translate the system-specific, real-world, system relationships into the fuzzy-associative-memory mathematical relations as well as the mathematical relation attributes that are required to support the effective aggregation of the fuzzy membership functions.

II. PROBLEM DEFINITION

The creation of system architecture as well as the evaluation, documentation, communication and development of the selected system architecture can be a highly challenging task when a system under development is relatively static with few interactions and interfaces. When the system under design is highly dynamic and contains a large number of context-specific, adaptable interfaces, the task of system architecting can become overwhelming for even the best system architects. Evolutionary algorithms and computational intelligence techniques are identified as the basis of tools and techniques that can be used by system architects to greatly increase the probability that successful system architectures will be designed, developed, and deployed.

System architecting is a professional set of tasks associated with the creation of large-scale systems for a given customer. These tasks are described in the framework of three basic roles. These roles are that of the customer, the system builder and the system architect. The architect is viewed as an agent of the customer and guides the system creation in a manner that maximizes the benefit to the customer. The activity of system architecting is further defined by establishing a general system architecture development context. This general context is divided into six specific context areas; mission context, mission functions, system functions, system physical architecture, risk context and affordability context. While the system architect, acting as the customer agent, has the total responsibility for integrating the values and perspectives from all context views, the system architect only has controlling interest in the risk and affordability context. The system customer has the controlling interest in the mission context and the mission functions context, while the systems engineer has controlling interest in the system functions context and the system physical architecture context. These six contexts and their associated role-context groupings are shown in Figure 1, Architecture Development Context.

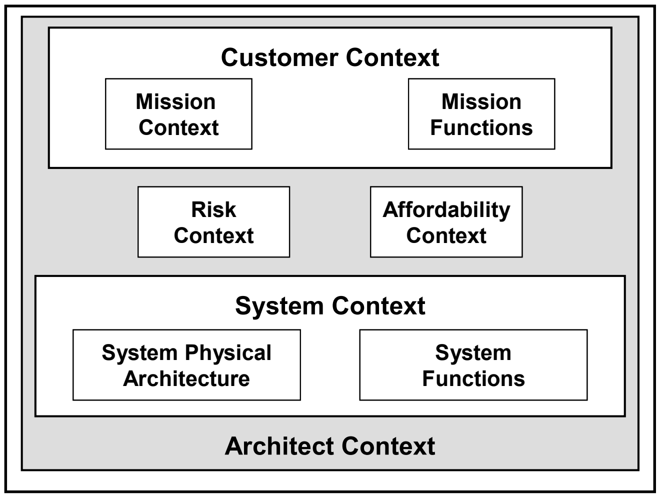

Figure 1 - Architecture Development Context

All system architecting tasks are based on a phased systems development approach, and are usually associated with one or more controlling system development lifecycle models. The specific architecting tasks will change in content and nature as a candidate system is moved through the selected system life cycle phases [1].

A. Mission and System of Systems Context Representation

Each of the primary mission and system-of-systems context areas shown in Figure 1 must be effectively represented in a form that clearly communicates the controlling context information to every person that has a primary role in the development of the system artifacts. The same context and interface information also must be translatable into an information form that supports the utilization of evolutionary algorithms and computational intelligence methods. Classical, structured systems engineering techniques provide the basis for the development of these structured forms of information. The concept of an abstract relation type (ART) has been developed to make a clear, direct connection between the classical forms of system representation and the information and data forms that are required to use evolutionary algorithms and computational intelligence techniques [2].

In the early stages of the system lifecycle, many critical design decisions and system-deployment value judgments must be addressed. However, as the current environment becomes more highly populated with existing systems that are candidates for inclusion into the system and/or system of systems under design, these early lifecycle decisions become much more difficult due to the very large number of possible interface connections. In addition to a large number of possible connections, the uncertainty associated with the behavior of each of the system configuration permutations creates a daunting task for the system-of-systems architect. Computational intelligence techniques have been proposed to help the system-of-systems architect in addressing this complex system behavior analysis task [3].

While the two previously mentioned references address a specific, local aspect of the system-of-systems evaluation problem, a global system-of-systems evaluation approach is necessary to support the integrated analysis of all aspects of any identified system-of-system solution. The classical system engineering methods associated with systems evaluation using measures of effectiveness has been adapted to support the global integration and evaluation of all three system architecting roles as they are integrated across the six given development contexts. The customer, with support from the system architect, develops and defines the required system mission functions as well as the lifecycle mission context. In essence, the customer guides the description of what needs to be accomplished under what conditions. The "what needs to be accomplished" portion of this activity generates the mission functions statement, while the "under what conditions" portion generates the mission context statement. Taken together, the two statements provide a high level problem statement for the created system solution to address.

Given both the statement of the mission functions and the mission context, the architect can then start the search for an acceptable system or system-of-system solution to the provided system problem statement. During this portion of the system architect's activities, a preliminary set of possible system solutions are generated and evaluated. Figure 2 shows the general components that are combined to create a metric for measures of effectiveness.

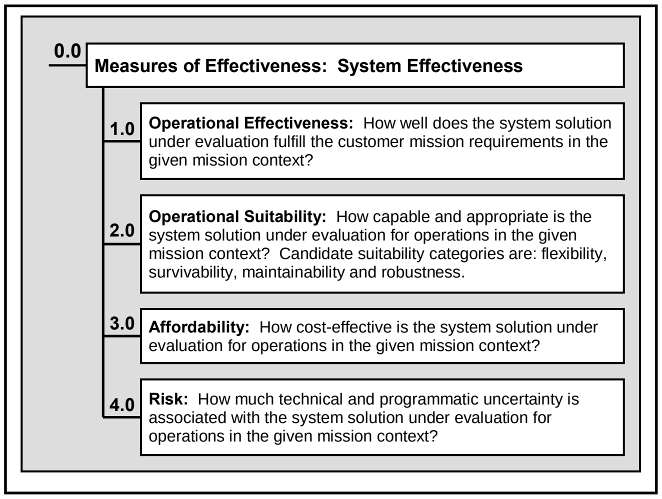
0.0 Measures of Effectiveness: System Effectiveness
1.0 Operational Effectiveness: How well does the system solution under evaluation fulfill the customer mission requirements in the given mission context?
2.0 Operational Suitability: How capable and appropriate is the system solution under evaluation for operations in the given mission context? Candidate suitability categories are: flexibility, survivability, maintainability and robustness.
3.0 Affordability: How cost-effective is the system solution under evaluation for operations in the given mission context?
4.0 Risk: How much technical and programmatic uncertainty is associated with the system solution under evaluation for operations in the given mission context?

Figure 2 - Measures of Effectiveness

B. Mission Function and System Function Representation

As detailed previously, one component of the measures of effectiveness metric is operational effectiveness. The operational effectiveness component is considered a controlling part of any system or system-of-systems evaluation. If the current system under evaluation does not perform the mission functions required by the customer, then the operational suitability, affordability and risk components are not relevant to the current customer mission. However, once a given system passes the initial operational effectiveness evaluation the other components of the measures of effectiveness can be used to rate and rank multiple system solutions with the same operational effectiveness score according to operational suitability, affordability and risk.

The customer mission functions and a candidate set of system functions must be compared to determine how well the system functions under evaluation accomplish the stated mission functions. To facilitate the evaluation of multiple system architectures, a standard hierarchical decomposition is used for both types of functions: mission functions and system functions. This approach has two primary benefits. The first benefit is the creation of a clearly identifiable layered hierarchy which supports the communication of distinct levels of functional decomposition. The second benefit is support for the establishment of well defined "functional closure" rules at every layer of the decomposition. These closure rules are simple and powerful. The basic premise of the closure rule is that every level in the functional hierarchy describes the same global function. The only difference between the functional layers is the amount of detail that is provided about the given, top-level function. At the highest level of a functional decomposition hierarchy, only the top-level function name is listed. At the layer below the top layer (level 2), all functions are listed that are required to accomplish the top level function. At the next level down (level 3), each of the functions necessary to perform the functions listed at the second level are given. This process of functional decomposition continues until the system architect determines that the system has been described at a level that supports the accomplishment of the customer architecture design. Since the mission functional decomposition and the system functional decomposition use the same basic process (though applied to different domain spaces), a generic functional decomposition is shown as an example in Figure 3, Generic Functional Decomposition Example.

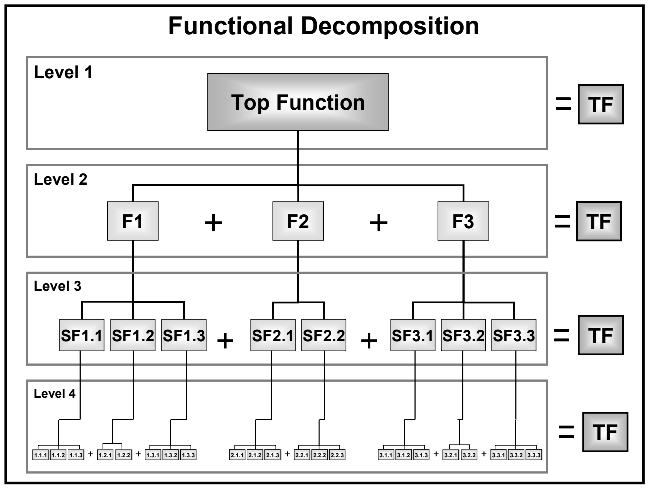

Figure 3 - Generic Functional Decomposition Example

Any candidate solution system for the customer can be physically decomposed into constituent subsystems, components and parts. Each physical system component can then be evaluated to determine the functionality provided by the specific component. The system physical architectural decomposition is considered next.

C. System Physical Architecture Decomposition

There are many different views and types of system architecture, including physical architecture, logical architecture, operational architecture and functional architecture. In many cases, a clear distinction is not made between or among these different types and views of system architecture. In this paper, the term physical architecture is used to describe the physical components that are arranged together to create the physical system that performs the system functions. The process of physical decomposition is almost identical to the process of functional decomposition. However, these two different views of a system, functional and physical, provide two different decomposition logics which then form two different types of hierarchical forms. In the process outlined in this paper, the system functions are evaluated against the customer mission functions. Then the candidate, physical-system components are evaluated to determine the functions performed by these physical components, both as single items or when multiple copies of the same item are used in combination

The complete physical system architecture context is then evaluated to determine how well the system functions performed by the candidate physical architecture fulfil the mission functions that have been articulated by the system architect acting as the customer representative. If a physical architecture gains a passing score in the operational effectiveness area then the other areas of operational suitability, affordability and risk can also be evaluated.

D. Evolutionary Algorithm Structure and Composition

Evolutionary computing techniques using crossover, mutation and selection operators are combined with fuzzy associative memories to create a hybrid computational intelligence algorithm that is applied to system-of-systems architecting tasks using the given set of measures of effectiveness The fuzzy associative memory is populated with a set of fuzzy rules in the form of the fuzzy generalized modus ponens. The fuzzy inference system encoded in the fuzzy associative memory effectively maps real world system architectural relationships between the fuzzy input variables and the fuzzy output variables. The generalized fuzzy modus ponens has three distinct components. These three components are, fuzzy rule, fuzzy fact and fuzzy conclusions. The fuzzy rules are developed using information and data collected from system architecting experts as well as the encoding of system architecting heuristics that have been developed over a period of time by the system architecting community. The outline of the computational intelligence algorithm is shown in Figure 4.

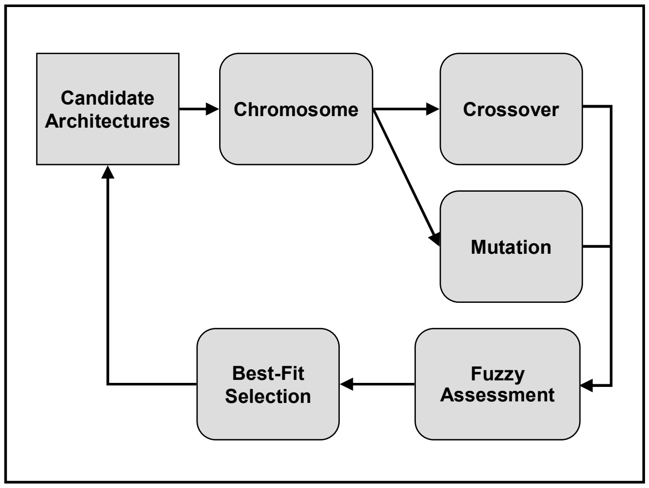

Figure 4 - Evolutionary Algorithm Outline

III. ARCHITECTURE GENERATION AND EVALUATION

The architecture generation activity starts with the system architect working with the customer to develop a set of mission functions that are explained in sufficient detail to support the documentation of the mission operational context and mission function context. The system architect then works with the systems engineer to create the required set of alternative physical system architecture and system function context views. One set of views is generated for each system architecture that will be evaluated as a solution for the customer system.

The lowest level of mission function decomposition was selected as the basis of the evolutionary algorithm chromosome genotype representation. While the mission functional context was chosen to represent the system operational phenotype as a set of 21 real numbers that arranged in an array structure. The first fifteen numbers provide a representation of how well the physical system architecture responds to the customer mission requirements. The last six numbers provide a representation of how well the robustness, reliability, availability, flexibility, survivability and affordability portions of the measure of effectiveness are addressed. Figure 5, Genotype to Phenotype Mapping, provides a graphical representation of this mapping relationship. The evolutionary algorithm is applied as shown in Figure 4, with the parents and children from each generation being placed in a common pool and the best individuals from the pool being selected as parents for the next generation of solutions. The system physical architecture algorithm has the following general steps:

- Generate 100 random individuals for the initial population
- Apply value mapping to determine component values
- Create subgroups of 20 individuals each
- Apply different crossover operators on different subgroups
- Apply mutation operator
- Evaluate and select best groups
- If acceptable solution is found, STOP
- Generate new population
- GOTO step two.

This general procedure can be modified to apply varying weights to different parts of the calculation as well as applying a set of minimum value criteria that filter out individual solutions that do not meet the minimum operational value.

The approach can be changed by modifying the number of system and mission functions that are represented by the chromosome. The hierarchical nature of the mission function representation provides a mechanism to specify the system level of abstraction that is being evaluated. As this type of technique is applied over a wide range of mission and system problem spaces sets of typical solution system configurations can be identified and cataloged.

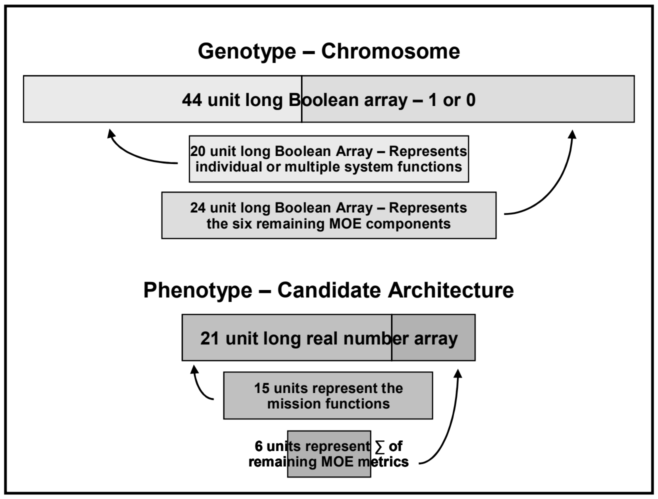
Genotype – Chromosome
44 unit long Boolean array – 1 or 0
20 unit long Boolean Array – Represents individual or multiple system functions
24 unit long Boolean Array – Represents the six remaining MOE components
Phenotype – Candidate Architecture
21 unit long real number array
15 units represent the mission functions
6 units represent ∑ of remaining MOE metrics

Figure 5 - Genotype to Phenotype Mapping

IV. SYSTEM RELATIONSHIP DEFINITION, EVALUATIONS AND AGGREATION

A fuzzy associative memory is used as the best fit function in this evolutionary algorithm to select the fittest individuals from the population of generated solutions. The system physical architecture is represented by two primary components: the system functions and the system suitability attributes. The six system suitability attributes are mapped directly into two of the measures of effectiveness sub-components: operational suitability and life cycle cost. One key observation is the system suitability measures apply to the physical system architecture, not the mission function hierarchy. The connection between the mission function hierarchy and the physical system architecture is the system function hierarchy. In general, the mapping between the physical system architecture and the system functional hierarchy will set the basic mapping that is used to determine the fuzzy performance values associated with the sub levels of the mission function hierarchy.

The affordability value will be discussed first as it is a value that is determined by summing the life cycle cost of all of the physical system architecture components. In general, if life-cycle cost for each of the sub-systems that perform the system functions used to execute the mission functions is low, then the affordability will be high. If the life cycle cost of each of the sub-systems that perform the system functions that are used to execute the mission functions is high then the affordability will be low. This example of the affordability performance measure demonstrates a logical value relationship that flows from the idea of physical composition used to construct the physical system architecture.

The other five architectural suitability measures have the same structural connection from the physical system architecture through the system function hierarchy to the mission function hierarchy. However the robustness, reliability, adaptability, flexibility, and survivability fuzzy performance measures are not independent. As a result, these fuzzy performance measures may depend on each other in some manner that may be application-domain dependent. While significantly computationally different from the classical system effectiveness measures, the fuzzy performance measures are designed to achieve the same quantifiable measures of system effectiveness.

The remaining five fuzzy performance measures are evaluated in two groups. Group one contains robustness and survivability, while group two contains reliability, adaptability and flexibility. Robustness and survivability depend a great deal on the given design mission profile. The design mission profile will give all environmental, threat, and operational parameter values and performance expectations. So, the survivability values will be given in the context of environmental, operational and other active threats detailed in the mission profile. The robustness fuzzy performance measure will indicate the ability of the selected physical system to operate at the margins of, or outside, the operational margins given in the design mission profile. Similar to the affordability metric, group one fuzzy performance measures are not necessarily additive or linear. Each sub-system could be very robust and highly survivable, but the integrated system segment or total system could be fragile and highly vulnerable. Therefore, the physical system architecture must be evaluated at each level of physical integration to assure that the robustness and survivability fuzzy performance measures are being properly addressed.

Similar to the group one metrics, the group two fuzzy performance measures - reliability, adaptability and flexibility - are not necessarily additive or linear. Unlike the group one metrics, the group two metrics are associated directly with the configuration of the physical system architecture, and how this architecture is used to perform the system functions that support the execution of the mission function.

For example, a physical system could be architected in a manner that assigns one physical system segment to provide system functions that support the execution of mission functions A, B, C and D. Another physical system segment could be assigned the task of providing the system functions that support mission functions D, E, F and G. In that instance, all mission functions except mission function D are single points of mission functional failure. This type of physical system architecture may have adequate system reliability but would have low adaptability and flexibility values. Given the same two physical system segments, and the ability of each physical system segment to adapt concurrently to provide support for each mission function in two different ways, then the physical system architecture becomes much more reliable, flexible and adaptable. Further, if the physical system has segments that are flexible and adaptable to support a general class of mission system functions no matter what the specific mission functions are, then the physical system architecture becomes even more flexible and adaptive.

As mentioned earlier in this paper the generalized fuzzy 'modus ponens' is used in fuzzy logic, and has three distinct parts: fuzzy rule, fuzzy fact and fuzzy conclusions. An example is:

If **A** is **L**, then **B** is **M** (fuzzy rule) (1)
A is **X** (fuzzy fact) (2)
B is **Y** (fuzzy conclusion). (3)

In this case the fuzzy rule can be written in terms of the fuzzy relation, *FR*, as:

$$(\mathbf{A},\mathbf{B}) \text{ is } FR \tag{4}$$

where *FR* represents the fuzzy implication relation.

The fuzzy conclusion is generated using the compositional rule of inference:

$$\mathbf{M} = \mathbf{X}\ \boldsymbol{c}\ FR \tag{5}$$

where ***c*** indicates fuzzy composition.

One perceived problem with this type of knowledge encoding is that the number of rules required to correctly represent a specific knowledge area is the square of the number of antecedent rules or conditions found in the knowledge area. The classical approach to rule-based systems, both crisp and fuzzy, is the use of the logical "and" operator which creates a series of valid intersections for a given rule. This is called the intersection rule configuration (IRC). A typical rule is given as:

If (**a** and **b**), then **z** (6)

If either **a** or **b** changes, the relation to **z** changes as well.

As more antecedent elements are added to the equation, the required intersection definitions become a more complex task. The union rule configuration (URC) was developed to help address the concerns associated with the classical intersection approach. The URC equation [4] is produced from the IRC [(**a** and **b**) then **z**] using Combs formal propositional logic transforms, giving:

(**a** then **z**) or (**b** then **z**) (7)

This approach is used to reduce the number of rules associated with the fuzzy inference system. The URC models a real world relationship without addressing specific characteristics of the fuzzy mathematical relation that represents the real world relationship. The three properties of relations are symmetry, transitivity, and reflexivity [5]. The symmetry set contains three basic properties: symmetry, asymmetry and non-symmetry.

- Symmetry can be identified by the following characteristics, "If any object projects the relation to a second object, then the second projects it to the first object."
- Asymmetry is defined as, "If any object projects the relation to a second object, then the second object does not project the relation to the first object."
- Non-symmetry is a relation that is neither symmetrical nor asymmetrical.

Another key set of properties associated with relations is the transitivity set. This set contains transitivity, intransitivity and non-transitivity.

- Transitivity is identified by the following characteristics, "If any object projects this relation to a second object and the second object projects the realation to a third object, then the first object projects it to the third object."
- Intransitivity is defined as, "If any object projects the relation to a second object and the second object projects the relation to a third object, then the first object does not project the relation to the third object."
- A non-transitive relation is a relation that is neither transitive nor intransitive.

The third and last set of relation properties is the reflexivity set which contains three properties: reflexive, irreflexive, and non-reflexive.

- The reflexive property is defined as "Every object projects the relation upon itself."
- A relation is irreflexive "No object projects the relation upon itself."
- A relation is non-reflexive if it is neither reflexive nor irreflexive.

When the controlling relationships that govern the development and evaluation of the fuzzy membership functions are transitive, symmetric and reflexive then the fuzzy membership values may be aggregated, or summed, in a manner that preserves the value of the real world relationship that is being mapped by the fuzzy associative memory. Each type of real world relationship must be evaluated in the context of the current application to determine if the use of the URC aggregation operator is warranted.

V. SUMMARY AND CONCULSIONS

Evolutionary algorithms and computational intelligence techniques have been applied to support a wide range of analysis and evaluation tasks associated with system and system of systems architecting. Combined with classical structured system engineering techniques a evolutionary algorithm that uses a fuzzy associative memory as a best fit functions was developed and applied to a sample system of systems architecture generation and evaluation task. This example system of systems architecture generation and evaluation activity was considered successful and highlighted a number of areas where further research and development will be needed to discover the most effective application approaches. One such area is determining the best methods for encoding system physical and functional structural information. Another area is the evaluation and exploration of a set of system relationships that can be aggregated using the URC aggregation methods of fuzzy composition.

Computational intelligence techniques present a very promising area of research and development associated with the design, engineering and development of complex system architectures.

REFERENCES

[1] M. W. Maier and E. Rechtin, *The Art of System Architecting*. CRC Press, Boca Raton, Florida, 2002.

[2] J. J. Simpson, C. H. Dagli, A. Miller, "System Evaluation and Description Using Abstract Relation Types (ART)." CD Proceedings of 1St Annual 2007 IEEE Systems Conference, Honolulu, Hawaii, April, 2007.

[3] C. H. Dagli and N. Kilicay, "Understanding Behavior Of System Of Systems Through Computational Intelligence Techniques." CD Proceedings of 1St Annual 2007 IEEE Systems Conference, Honolulu, Hawaii, April, 2007.

[4] E. Cox, *The Fuzzy Systems Handbook, Second Edition.* Appendix A, pp. 659-680, AP Professional, New York, 1999.

[5] H. Pospesel, *Introduction to Logic: Predicate Logic*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1976.