Genomic Systems Design: A Novel, Biologically-Based Framework for Enhancing the Adaptive, Autonomous Capabilities of Computer Systems

William E. Combs
The Boeing Company
P.O Box 3707
M.S 7P-13
Seattle, WA 98124
william.e.combs@boeing.com

Jeffrey J. Weinschenk
University of Washington
Dept. of Electrical Engineering
253 EE/CSE Building
Campus Box 352500
Seattle, WA 98195-2500
jjw77@ee.washington.edu

Robert J. Marks II
Baylor University
Dept. of Engineering
Waco, Texas 76798
r.marks@ieee.org
Robert_Marks@Baylor.edu

***Abstract*— Genomic Systems Design (GSD) is an outgrowth of the Union Rule Configuration (URC), a propositional logic construct that eliminates the combinatorial problem for rule-based systems. Its architecture is scalable, adaptive and fault-tolerant and is well-suited to multi-criteria decision systems and applications that must deal with sparse and missing data.**

This novel programming paradigm is similar in architecture to a biological process called symbiogenesis. This biological process is said to facilitate the evolution of new species through the inheritance of genomes from organisms that are participating in symbiotic relationships. This similarity, together with the characteristics of the URC, enables Genomic Systems Design to offer a promising alternative methodology for the design of autonomous agents/robots, fault-tolerant and adaptive control systems, cellular automata and bioinformatics.

I. Introduction: The Evolution of Genomic Systems Design

Rule-based systems often form the computational structure for artificial intelligence, fuzzy logic, security access systems and an ever-growing number of business systems which use rules to avoid hard coding business logic. However, rules in the form **[(P intersection Q) implies R]** suffer from a combinatorial problem in that the rules increase *exponentially* as antecedents are added, severely impacting performance and robustness.

Combs [2, 3] designates this type of rule construct as the *Intersection Rule Configuration* (IRC) to emphasize that its rules are generated by intersecting propositions. Combs developed an equivalent *scalable* logic structure **[(P implies R) union (Q implies R)]** in which rules increase *linearly* as antecedents are added, and calls this alternative structure the *Union Rule Configuration* (URC) to emphasize that rules are created through the union of functioning implications. Weinschenk *et al.* [15, 16, 17] formalized URC design by showing its association with neural networks and proving that a layered URC network is a universal approximator.

The URC can also model biological genomes: [(Gene-1 implies R) union (Gene-2 implies R) union . . .]. In a recent book *Acquiring Genomes: A Theory of the Origins of Species*, Lynn Margulis and Dorion Sagan refer to a process called *symbiogenesis* [12] that they maintain is the principal evolutionary mechanism for the multiplication of species. According to their theory, new species are the result of the inheritance of genomes from different organisms that are participating in symbiotic relationships – a process that closely parallels the additive characteristics of the URC. This process is recursive and also provides a robust framework for adaptive learning.

The URC enables the development of highly adaptive, fault-tolerant computer systems through the additive and emergent characteristics of the union operations that knit the various functioning implications into a synergistic whole. However, the original IRC/URC designations tended to limit the perceived scope of these propositional logic structures to rule-based systems even though the IRC mindset also dominates such areas as expert systems, decision trees, belief networks, Bayesian inference, neural networks and control systems.

In an effort to overcome this perceived limitation, we have designated the URC's capabilities that address a more generic systems design methodology as *Genomic Systems Design* (GSD), a moniker that also emphasizes the similarity the URC structure has with the biological counterpart mentioned above.

In this paper, we briefly address the characteristics of GSD. In Section II, we define symbiogenesis and show how GSD can model its behavior in a high-level manner. In Section III, we share several important characteristics of the URC that make possible the robust nature of GSD. And in Section IV, we conclude by outlining several beneficial areas

for GSD.

It should be understood at the outset that GSD is not a new methodology but rather a more generic designation for the URC. The characteristics listed in Section III, together with many of the areas of benefit listed in Section IV have been demonstrated previously both through IEEE publications and conference presentations and through presentations within Boeing. Its association with symbiogenesis is also meant to show how the latter's behavior can be modeled through the programming characteristics of the URC.

II. A Brief Overview of Symbiogenesis

The next five paragraphs outline a recursive biological pattern that facilitates symbiogenesis:

The cells of an individual in a given species will specialize from stem cells to various cell types during the course of maturation from conception to adulthood [5, 6, 7]. This process is governed by the genome residing in every cell as the cell relates functionally to its environment.

Each specialized cell type functions in a certain manner and the cells comprising each type strive to fulfill the goal of their functional role in balance with the goals of the cells of the other cell types.

Extended stress on this individual indicates that even with the most delicately balanced relationships among its specialized cell types, the cells still cannot achieve their functional goals satisfactorily, jeopardizing the survivability of the individual.

In such a stressful situation, the individual will more likely enter into a symbiotic relationship with an organism of another species in order to reduce its stress and improve its ability to survive.

Under certain circumstances, this symbiotic relationship can result in the creation of a new species through the merging of the genomes of the two organisms, a process called symbiogenesis [12].

GSD will model this recursive pattern by incorporating several important elements outlined in the next four paragraphs:

In a traditional programming paradigm, the methods of lower-level objects are called by the methods of higher-level objects within the context of code that explicitly governs their behavior. In order to change behavior, it is often necessary to modify this code. As an alternative, a higher-level GSD method will invoke lower-level methods as though they were running on separate threads to simultaneously and collaboratively achieve the goal(s) of the higher-level method. To illustrate, in the classic Artificial Life *BOIDS* example [14], the various components of flocking behavior, represented by the following criteria, can be invoked to function simultaneously and collaboratively so that the flocking behavior emerges from their interaction: (1) Collision Avoidance: avoid collisions with nearby flock-mates; (2) Velocity Matching: attempt to match velocity with nearby flock-mates; and, (3) Flock Centering: attempt to stay close to nearby flock-mates.

The goal of each higher-level object will also act as a type of training algorithmic objective function, enabling the lower-level objects to learn from and adapt to their environment as they align with the higher-level objective function. For example, as indicated above for flocking, there would be no higher-level code that specifically programmed a bipedal robot to walk since walking is an emergent, macro-behavior. Instead, multiple lower-level implication relations with objectives like balancing, mirrored symmetry and response to threshold stimulus, would give the bipedal robot the initial "genetic" ability to walk. A training algorithm functioning through the objective functions of higher-level objects would further train and adapt the robot as it takes its first steps so that it learns to walk more efficiently and effectively.

Depending on the environment, each implication relation will be able to perform its task with relative, measurable success. The union operations will enable these relations to share their relative success with each other so that a synergistic balance can be manifested in the higher-level objects.

Degrees of stress on the system will be measured as the cumulative degree to which the implication relations can achieve their objectives. Training and adaptation will be used initially to respond to increasing degrees of environmental stress in an effort to achieve more desired objectives within the bounds of the system's domain space as defined by its "genome". However, the cumulative, diminishing success will reach a threshold that alerts the system architect to develop an appropriate implication relation(s) – a symbiotic partner(s) – to help overcome the effects of the stress. Later work will focus on additional methods to achieve this "evolutionary" acquisition.

Symbiogenesis also provides a robust environment for adaptive learning. That is, an agent in a situation that inhibits its ability to achieve its goals can use a threshold signal to acquire, develop or adapt logical rules, algorithms or relations that will improve its ability to achieve its goals.

III. Characteristics of Genomic Systems Design

Here we share several of the most important characteristics of the URC that enable the robust nature of GSD.

A. Coupling and Cohesion

Coupling is the strength of the relationships between

modules. *Cohesion* is the strength of the relationships among the components of one module. System robustness is improved whenever coupling can be reduced and cohesion increased [1]. For the IRC, P and Q do not have an independent relation with R, so cohesion is low. And since P must intersect with Q in order to produce a relation with R, coupling is high. For the URC, P and Q each have their own relation with R, so cohesion is high. And, since changes in P do not affect Q, coupling is low.

Loosely coupled, highly cohesive systems are also easily modified. Conversely, it is often so expensive to update a large, tightly coupled system that short-cut solutions and "temporary" patches are employed, leading to brittle and error-prone applications that must eventually be replaced.

B. Scalability: The Combinatorial Problem

As stated above, most traditional rules are in the IRC format [(P intersection Q) implies R]. Although IRC rules are the more familiar rule construct, they create a difficult combinatorial problem. Since P and Q are coupled to each other through intersection, the number of values in P must be multiplied by the number of values in Q in order to obtain all possible solution options. (Obviously, all possible solution options are not always necessary.) So, if there are five values for each input variable, then the maximum number of rules for a single input, single output system would be 5. For a two input, single output system, the maximum number of rules would be 25 and a three input, single output system would yield a maximum of 125 rules – *an exponential increase*.

The equivalent URC rule architecture [(P implies R) union (Q implies R)] eliminates this combinatorial problem. Since P and Q each have their own independent relation with R, changes in the value of P do not affect Q's relation with R. As a result, instead of multiplying the number of variable values with each other to obtain the maximum potential solution options, URC rules only require that the number of values be added. So, instead of the exponential increase shown above, a single input, single output system would have a maximum of 5 rules. For a two input, single output system, the maximum number of rules would be 10 and a three input, single output system would yield a maximum of 15 rules – *an additive increase.* We have shown that the URC radically reduces the computational time for complex data sets and also virtually eliminates the need for pruning strategies to improve the computational efficiency of rule-based applications [3, 4].

C. Fault Tolerance

Fault tolerance is a vital dimension in any mission critical system because it diminishes the adverse affects of defects that might otherwise impair the system's functionality. It is especially helpful in those situations where an unexpected fault could jeopardize or severely impair the success of the mission.

While defect reduction through such processes as Six Sigma [18] is a necessary component of software reliability, it may do little to guard against in-service malfunctions and accidents. In some instances, component redundancy is employed to provide a kind of back-up fault tolerance. But this option usually adds weight and complexity to the mission.

The overlapping region in Figure 1 indicates that these implication relations are both capable of performing certain functions – from different perspectives. On the one hand, if both relations perform these overlapping functions, the union operator will generate the functionality for the application. On the other hand, if one relation fails to perform one of these functions, the other one will still feed the union operator so that the functionality will continue to be generated for the application. So, the two implication relations (agents) can provide fault tolerance for each other.

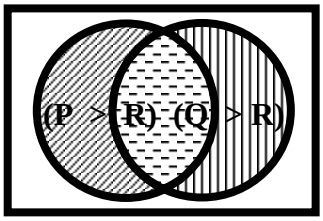

Fig. 1. Venn diagram of overlapping UNION implication relations

Stated another way, the circle encompassing (P > R) represents its primary objective. The portion of that circle intersected by the circle surrounding (Q > R) represents the latter's ability to perform that task as well. The URC's higher-level functionality represented by the union operator will give both implication relations the task to perform. (Q > R) will monitor (P > R)'s ability to perform its primary task to see if its performance falls below a certain goal threshold. (Q > R) will act as a symbiotic partner, performing this task to the extent that it perceives (P > R) is failing and to the extent that it has sufficient capacity.

For example, consider a four-wheeled robot in which each wheel can act independently. Further assume that the primary objective for the front pair of wheels is navigation while the primary task of each wheel is propulsion. Based on its configuration, we can say there are actually three ways to turn the rover either right or left. First, the implication relation controlling the front pair of wheels can perform the task. But the implication relation controlling the rear wheels operating in tandem can also turn the rover. And, if the rover needs to turn to the right, the implication relation controlling the left wheels can rotate them faster than the right wheels.

Assume that a higher-level implication relation needs to

turn the rover to the right and gives all three of these lower-level relations the task to perform. Since the task is the primary objective of the relation for the front pair of wheels, the other two relations can act as symbiotic partners monitoring the former to see if its performance falls below a certain goal threshold. If it does, then these two partners can assist in executing the task to the extent that the primary relation is failing and to the extent that they have sufficient capacity.

D. Adaptability

The enclosure around the implication relations (agents) in Figure 2 represents the environment in which they function.

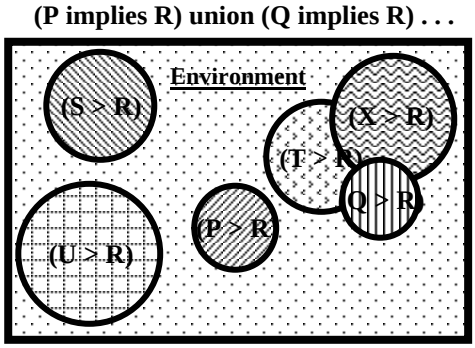

Fig. 2. Venn diagram of the URC in a given environment

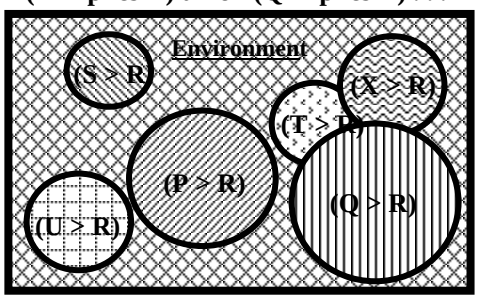

Fig. 3. Venn diagram of the URC in a different environment

As the environment changes (Figure 3), the relationship each agent has with the environment changes as well. That is, one type of environment will favor the functionality of certain agents more than others based on each agent's perspective.

These changes in the functionality of the individual agents modify the ultimate output of the system resulting from the union of their functionalities. This modification allows the application to dynamically adapt to the changes in its environment without altering its code or employing parameters.

E. Specialization

In Section II, we referred to the specialization of cells for an individual in a given species. In an ant colony, each ant possesses the colony's genome enabling its behavior to be specialized to a number of genetically-defined roles (nest maintenance, foraging, midden work, etc.) as it responds proximally to various environmental stresses in collaboration with its other nest-mates [8]. Through its implication relations, the URC can model specialization whether it is manifested in a single agent or in a team of agents working together to perform some mission.

F. Emergence

"In these systems, agents residing on one scale start producing behavior that lies one scale above them: ants create colonies, . . . The movement from low-level rules to higher-level sophistication is what we call emergence" [10]. The important aspect of this process is that, on the one hand, the lower-level relations contribute to the higher-level emergent functionality. But on the other hand, the lower-level relations can neither fully determine not predict the higher-level behavior. The URC is an emergent architecture in that the *union* of its lower-level relations [(P implies R), (Q implies R)] enables emergent functionality/behavior that lies one scale above them.

G. Sparse and Missing Data Resolution

In many applications, missing data is interpolated in some manner – a process that can bias the outcome. Estimating a missing data point is more easily accomplished when only a few values are absent. However, interpolation becomes far more complicated when the majority of values are missing.

We have shown that since the URC relations are connected by logical unions, a system can ignore any relation where the attribute value is null, zero or missing. So, sparsely populated records do not have to be doctored or deleted in order for an application to process them reliably [4].

H. Functionality-As-A-Premise

As shared earlier, the URC structure emphasizes that rules are created/extended through the union of functioning implications. Thus, functioning implications must exist **before** the union can take place. Symbiogenesis underscores that genomes likely evolved using a similar strategy. According to this theory, genes were added to the genome of an individual under stress based on the fact that the genes of its symbiotic partner were functioning before the merger in a manner that could relieve the stress.

On the other hand, the IRC structure generates a functionality-as-a-goal process. Its low cohesiveness results from the fact that its antecedent propositions (**P** intersection **Q**) do not enjoy independent, functional relations with their consequent proposition (**R**). Instead, the intersection operator yields another proposition (**I**) that does have a functioning relation with the consequent proposition (**I**

implies **R**). So, functionality is a goal of the IRC process.

I. Multi-Criteria Decision Making

We are most accustomed to UNION Venn diagrams as shown in Figure 1. However, Figure 4 is just as valid a representation of UNION. Moreover, it illustrates that the two elements do not have to overlap as they do for INTERSECTION in order to produce a valid result. This capability enables the URC to process multi-criteria decisions even when the perspectives are mutually exclusive.

(P implies R) union (Q implies R)

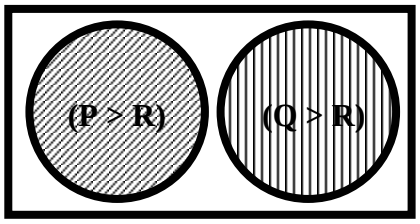

Figure 4 Venn diagram of non-overlapping UNION implication relations

IV. Areas of Benefit

A. Parallel and Distributed Processing

In addition to reducing computation time, the URC also provides a GSD framework with much more extensive parallel and distributed processing capabilities than might otherwise be possible since the relations are coupled by union. This means the relations [(P implies R), (Q implies R)] can be executed by separate processors either on the same computer or on different computers across a network.

B. Fault-Tolerant Control Systems

A fault-tolerant control system should be able to survive the malfunction of any one controller, should allow the application to degrade its performance gracefully and should maintain the degradation of the program's functionality within the scope of the controller's design envelope.

For any system that might experience the gain or loss of a number of inputs, agents, sensors or components during operation, the URC architecture facilitates the fault-tolerant characteristics of GSD because this variability is so easily managed.

C. Disparate Database Utilization

It is not uncommon for the databases of collaborating institutions to contain dissimilar attributes since each institution is likely to focus on a different aspect of the project. Unfortunately, the structure of these disparate databases can hamper utilization by the various participating researchers. One solution is to limit shared information to the attributes common to all institutions. However, this approach can restrict the overall potential use of the data. The same principle that allows the URC to work with sparse and missing data also allows a GSD project to combine dissimilar attribute data from disparate databases for use by the various participants.

D. Autonomous Agents/Robots

As shared earlier, we are most accustomed to UNION Venn diagrams as illustrated by Figure 1 even though the configuration in Figure 4 is just as valid. Through addition, the URC enables us to extend Figure 4 to Figures 2 and 3. By expanding those configurations, Figures 5 through 7 show the URC Venn diagrams morphing into the quintessential example of an autonomous, agent-based system.

(P implies R) union (Q implies R) . . .

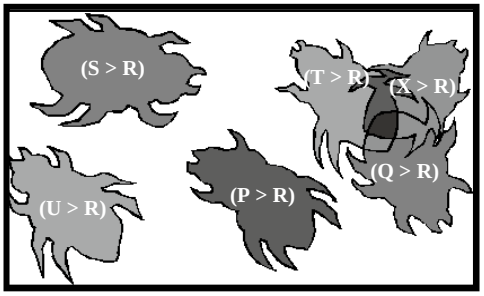

Figure 5 Venn diagram of the URC morphing . . .

(P implies R) union (Q implies R) . . .

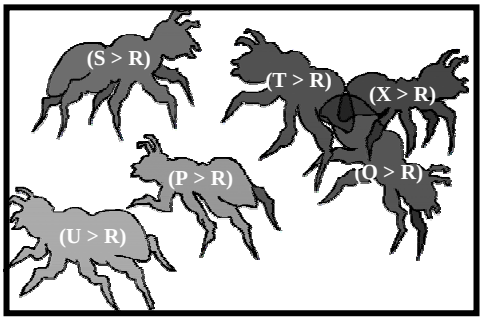

Figure 6 . . . into the quintessential example . . .

(P implies R) union (Q implies R) . . .

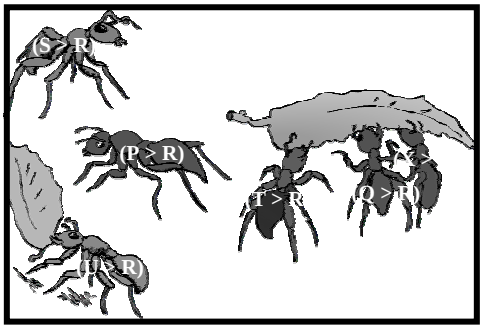

Figure 7 . . . of an autonomous, agent-based system

A current preferred method for managing the activities of teams of millibots is a hierarchical structure in which ever-larger robots oversee the activities of smaller robots [9]. As an alternative, these smaller robots could all possess the same "genome", specializing according to the needs of the mission and collaborating through their shared union operators. By functioning in this manner, similar to the members of an ant colony, agent-based millibots in a GSD configuration could eliminate the need for a hierarchy of larger robots to coordinate their activities.

E. GSD and Cellular Automata

Scientists have known for many years that populating cellular automata with rules can generate emergent behavior – a process also known as distributed emergent computation [11]. This effect was popularized by the Game of Life, invented in 1970 by British mathematician John Conway, now at Princeton University.

While cellular automata have long been used to provide the structure for proximal cellular relationships, they do not specify any configuration architecture for the rules governing cellular behavior. Since GSD can model genomes – the architecture for the genetic code within biological cells – GSD can also provide the configuration architecture for the rule sets within each cell of cellular automata.

F. The Genome Project (Next Steps)

Bioinformatics is a relatively new field that brings together biology, genetics and computer science to address the overwhelming challenges initiated by the Genome Project [13]. While the initial efforts of bioinformatics concentrated largely on the demands of the Genome Project, there is an even greater need now that the preliminary project goals have been achieved – gaining benefit from all this newly acquired knowledge.

The genomic structures that have been compiled do not explain how this genetic code manifests itself functionally in a particular species member. To this end, we need to model genomic functionality in order to better understand how a given genotype configuration influences the functionality of a particular phenotype of a species.

GSD can model the processes surrounding symbiogenesis. Given its ability to demonstrate how a genome could be developed, GSD might also offer a promising architecture for aiding in the understanding of how the various genes interact functionally with each other.

V. CONCLUSION

We have highlighted a novel programming paradigm that is similar in architecture to a biological process called symbiogenesis. This similarity, together with the strengths inherited from the URC, enables Genomic Systems Design to offer a promising alternative methodology for the design of autonomous agents/robots, fault-tolerant and adaptive control systems, cellular automata, and the next steps for bioinformatics.

ACKNOWLEDGEMENTS

We wish to thank Larry Bugbee, a friend and fellow Boeing employee of William E. Combs, for his suggested emendations to this paper.

REFERENCES

[1] E.B. Allen and T.M. Khoshgoftaar, "Measuring Coupling and Cohesion: An Information-Theory Approach", *Sixth IEEE International Symposium on Software Metrics*, Boca Raton, Florida, p. 119, 1999.

[2] W.E. Combs, "Reconfiguring the Fuzzy Rule Matrix for Large, Time-Critical Applications", *Third Annual International Conference on Fuzzy-Neural Applications, Systems and Tools* PennWell Publishing Company, 10 Tara Blvd., 5th Floor, Nashua, NH 03062-2801, Nov., 1995.

[3] W. E. Combs and J. E. Andrews, "Combinatorial rule explosion eliminated by a fuzzy rule configuration," *IEEE Trans. Fuzzy Systems*, vol. 6, no. 1, pp. 1-11, Feb., 1998.

[4] W.E. Combs, "Using Fuzzy Logic in Large, Complex Data Mining Applications", *IEEE World Conference on Computational Intelligence*, May, 2002.

[5] M.S. Gazzaniga, R.B. Ivry and G.R. Mangun, *Cognitive Neuroscience: The Biology of the Mind*, 2nd. Ed., pp. 400ff, W.W. Norton and Co., 2002.

[6] W.W. Gibbs, "The Unseen Genome: Gems Among The Junk", *Scientific American*, pp. 46-53, Nov., 2003.

[7] W.W. Gibbs, "The Unseen Genome: Beyond DNA", *Scientific American*, December, pp. 106-113, Dec., 2003.

[8] D. Gordon, *Ants at Work: How an Insect Society is Organized*, pp. 29ff, W.W. Norton & Co., 1999.

[9] R. Grabowski, L.E. Navarro-Serment and P.K. Khosla, "An Army of Small Robots", *Scientific American*, pp. 63- 67, Nov., 2003.

[10] S. Johnson, *Emergence*, p. 18, Scribner, 2001.

[11] E. Klarreich, "Computation's New Leaf: Plants may be calculating creatures", *Science News*, vol. 165, pp. 123-124, Feb. 21, 2004.

[12] L. Margulis and D. Sagan, *Acquiring Genomes: A Theory of the Origins of Species*, Basic Books, 2002.

[13] D.W. Mount, *Bioinformatics: Sequence and Genome Analysis*, Cold Spring Harbor Laboratory Press, 2001.

[14] C.W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model, in Computer Graphics", 21(4) *(SIGGRAPH '87 Conference Proceedings)*, 1987, pp. 25-34. See http://www.red3d.com/cwr/boids/

[15] J. J. Weinschenk, W. E. Combs, R. J. Marks II, "Avoidance of rule explosion by mapping fuzzy systems to a disjunctive rule configuration," *IEEE Int'l Conference on Fuzzy Systems*, St. Louis, MO, pp 43-48, 2003.

[16] J. J. Weinschenk, R. J. Marks II, W. E. Combs, "Layered URC fuzzy systems: a novel link between fuzzy systems and neural networks," *Proc. IEEE Intl' Joint Conf. on Neural Networks*, Portland, OR, pp. 2995-3000, 2003

[17] J. J. Weinschenk, W. E. Combs, R. J. Marks II, "On the avoidance of rule explosion in fuzzy inference engines," Submitted to IEEE Trans. Fuzzy Systems, Nov. 12, 2003.

[18] See http://www.isixsigma.com/