A Geometric Rule Extraction Approach Used for Verification and Validation of a Safety Critical Application

Marjorie Darrah[1], Brian Taylor[1], Michael Webb[2], Rhett Livingston[3]

[1]Institute for Scientific Research, Inc., [2]Lockheed Martin Technology Services, [3]ProLogic, Inc., Fairmont, WV 26554
mdarrah@isr.us, **btaylor@isr.us**, **michael.webb@lcmo.com**,**rhett@prologic-inc.com**

Abstract

This paper describes a geometric algorithm to extract deterministic rules from a dynamic cell structure (DCS) neural network and the rationale for extracting these rules. The DCS is a type of self-organizing map neural network that has been used in a real-time adaptive flight control application. The purpose for extracting rules in this instance is to determine whether such rules, along with other techniques, could be used in the verification and validation (V&V) of a neural network serving in a safety-critical role. This paper introduces a geometric approach to creating rules that mimic an instance of the trained DCS with 100% agreement. The paper will explain the intelligent flight control application of the DCS, describe the geometric method used for rule extraction, provide experimental results of the rule extraction techniques, and examine the relevance of the rules to the V&V process.

Introduction

Neural networks are members of a class of software solutions well suited for domains of non-linearity and high complexity that are ill defined, unknown, or just too complex for standard programming practices. For high assurance systems that utilize neural networks such as autonomous mission control agents, vehicle health monitoring systems, adaptive flight controllers, or nuclear engineering applications, proving safety and correct operation requires formal and rigorous approaches.

Testing the neural network with data similar to that which is used in training is a common method to verify that a network has adequately learned the input domain. In non-critical applications, such traditional testing techniques may constitute a valid approach for determining acceptance of the neural network solution. However, in safety- and mission-critical systems, this standard approach alone cannot accomplish the required formal process for certification.

The V&V challenge is further compounded by adaptive neural network systems that modify themselves, during operation. Traditional software assurance methods fail to account for systems that change after deployment.

We have investigated neural network rule extraction to determine its utility for the V&V of neural networks in safety- and mission-critical applications. Neural network rule extraction is a technique that translates the acquired knowledge and decision process of a trained neural network into an equivalent decision process represented as a set of rules. The rules are generally a set of if-then statements that may be examined by a human.

Our previous effort (Darrah, Taylor and Skias 2004) in developing a rule extraction technique for DCS resulted in what we refer to as 'human understandable' rules. These rules did not exhibit the high level of agreement with the DCS network that was desired nor were they deterministic, but their simplicity served to provide insight to a person examining the network. This paper presents our continued work in which a second rule extraction algorithm based on a geometric approach was developed. This new algorithm extracts rules that have 100% agreement with the neural network. The new rules are deterministic, but far more complex and better suited for use with other software testing tools used for V&V than for human inspection.

The Dynamic Cell Structure Neural Network and Intelligent Flight Control System

NASA Dryden Flight Research Center (DFRC) is conducting experiments in Intelligent Flight Control (IFC) with collaboration from the NASA Ames Research Center (ARC), Boeing Phantom Works, and the Institute for Scientific Research, Inc. (ISR). The first generation (GEN1) of the IFC program identified aircraft stability and control characteristics using neural networks, and used this information to optimize aircraft performance in nominal and simulated failure conditions.

The DCS neural network, a type of self-organizing map, is an online adaptive network used within GEN1 to learn and adapt during flight to account for aerodynamic changes, such as ones due to actuator failures. The DCS used in the IFC program has a long development history. It was originally developed by (Bruske and Sommer 1994), is a derivative of work by (Fritzke 1994) combined with competitive Hebbian learning by (Martinez 1993), and was chosen for the GEN1 system by NASA ARC (Jorgensen 1997).

Designed as a topology-representing neural network, the DCS's role is to learn the topology of the input space with perfect preservation. The DCS accomplishes this by learning the function that describes a map of the input space, represented as Voronoi regions[1], and storing this map as the neural network structure. The neurons within the neural network correspond to the reference vector (centroid) for each of the Voronoi regions. The connections between the neurons, c_{ab}, are then part of a Delaunay triangulation[2] connecting neighboring Voronoi regions through their reference vectors.

The output of the IFC DCS is formed from a recall function that computes an interpolation between two centroids within the network. Given an input stimulus, v, a reference vector is selected from among all centroids as the *best matching unit* (BMU). The BMU is simply the neuron whose weights are closest to v. Along with the BMU, a *second best matching unit* (SEC) is also found. The SEC is the closest neuron to v in the BMU neighborhood, defined as the neurons connected to the BMU through the Delaunay triangulation. The IFC DCS recall function then generates an output based upon the BMU and SEC.

The DCS algorithm consists of two learning rules, Hebbian and Kohonen. Hebbian learning updates c_{ab} between neurons a and b to reflect the topology (triangulation) of the input space:

$$\dot{c}_{ab} = \begin{cases} 1 & a \in [BMU, SEC] \wedge b \in [BMU, SEC] \\ \alpha \cdot c_{ab} & \alpha \cdot c_{ab} > \theta \\ 0 & \alpha \cdot c_{ab} < \theta \\ 0 & a = b \end{cases}$$

where the connection is a perfect fit of 1, if a and b are the BMU and SEC for the current training stimulus. The forgetting constant, α, is included to produce a weakening between a and b if they are not currently the closest to the stimulus, and θ is the edge threshold, a minimum acceptable connection strength in order for the connection to be considered valid. Kohonen learning is used to adjust the weight vectors, w, of the neurons. The change in the weight vectors is represented by:

$$\Delta w_{BMU_i} = \varepsilon_{BMU}(v_i - w_{BMU_i})$$
$$\Delta w_i = \varepsilon_{NBR}(v_i - w_i)$$

where ε_{BMU} is the BMU weight adjustment parameter and ε_{NBR} is the weight adjustment applied to the neighborhood of the BMU.

Along with the two learning rules for DCS, the DCS network also has a growing rule that will add new neurons into its structure to reduce overall error. This ability gives the DCS neural network the potential to evolve into many different configurations.

The DCS, like other online adaptive systems, is of special concern with respect to V&V. Developers must be cautious about expanding their use into safety- and mission-critical domains. Our research investigated how rule extraction techniques could assist in the V&V of the IFC system as well as other such neural network systems.

The Rule Extraction Technique

In previous research conducted at the ISR a basic rule extraction technique for the DCS neural network was developed (Darrah, Taylor, and Skias 2004). Figure 1 shows the basic algorithm for extracting human readable rules from the DCS.

```
Input:
Weight vectors of the DCS (centers of Voronoi region)
Best matching unit for each input
Output:
One rule for each Voronoi region center of the DCS
Procedure:
Train DCS on data set
Record BMU for each input used for training
Collect all inputs with common BMU
For each weight vector $w_i$ (center of Voronoi region and BMU)
        For each independent variable
        $x_{i, lower} = \min\{x_{i,j} \mid x \text{ has BMU} = w_i\}$
        $x_{i, upper} = \max\{x_{i,j} \mid x \text{ has BMU} = w_i\}$
Build rule by:
        Independent variable in $[x_{i, lower}, x_{i, upper}]$
        Join antecedent statements with AND
        Dependent variable = category
                OR
        Dependent variable in $[y_{lower}, y_{upper}]$
        Join conclusion statements with AND
```

Figure 1 Human Readable Rule Extraction Algorithm

The rules generated by this algorithm are human understandable and of two types depending on the data used for training and the function the neural network is performing. The first type of rule format is generated when the neural network is performing as a classifier. The independent variables are continuous real valued and the dependent variable is categorical.

```
Type 1:   IF input_variable $_1 \in$ interval $_1$ AND…
                    AND input_variable $_k \in$ interval $_k$
          THEN output = class$_j$
```

The second type of rule is generated in the case where both the independent and dependent variables are continuous real valued and the neural network is performing as a function approximator.

[1] Given a set of n points in the plane, a Voronoi partition is a collection of n convex polygons such that each polygon contains exactly one point and every point in a given polygon is closer to its central point than any other.

[2] The Delaunay triangulation is a dual graph of a Voronoi diagram that connects the centers of the Voronoi regions to their neighbors to form a triangulation of the plane (if no two points are cocircular)

Type 2: IF input_variable $_1 \in$ interval $_1$ AND…
AND input_variable $_k \in$ interval I_k
THEN output_variable $_1 \in$ interval $_1$ AND…
AND output_variable $_j \in$ interval $_j$

This original algorithm was tested on the Iris benchmark data (Fisher 1936), available through the University of California at Irvine, because it is a common data set used by authors of other rule extraction techniques. Sample rules generated from the Iris data have the form:

IF (SL >=5.6 AND <=7.9) AND (SW >=2.2 AND <=3.8) AND
(PL >=4.8 AND <=6.9) AND (PW >=1.4 AND <=2.5)
THEN Virginica

This algorithm was also used to extract rules from the DCS trained on the IFC flight data with sample output:

IF (mach >=0.78799 AND <=0.78945) AND
(altitude >=19860.484 AND <=19889.6526) AND
(alpha >=1.7003 AND <=1.8842) AND
(beta >=-0.029893 AND <=0.015156)
THEN (cza >=0.015062 AND <=0.019333) AND
(czdc >=0.22274 AND <=0.2287) AND
(czds >=0 AND <=0)

The rules extracted using this process were human understandable but did not have the desired accuracy and did not give a deterministic output for each input. Testing of the rules against the Iris trained DCS showed an 82% agreement.

Refining the Algorithm

The objectives for refining the DCS rule extraction algorithm are as follows:

1. Increase agreement with the neural network
2. Provide deterministic rules that could be input to another V&V tool or implemented as a rule based system
3. Maintain human understandability

It was determined that two separate algorithms would be necessary to achieve both human understandability and determinism. A new algorithm was developed to generate deterministic rules. This algorithm utilized the structure of the DCS knowledge by capturing the Voronoi regions that partition the input space.

The Vornoi regions are convex polygons in two dimensions and convex *n*-dimensional polyhedra in *n* dimensions. The original rule extraction algorithm did not capture the entire polygon or polyhedron region with the rules. In essence, it developed rules that used a box to approximate the region. (See Figure 2)

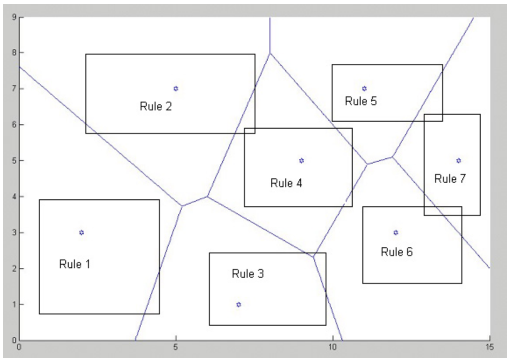

Figure 2 Original Rule Coverage

The new rule extraction algorithm was developed to completely capture the polygonal regions of the input space that represent the structure of the trained DCS. Figure 3 below shows a two-dimensional example of how the new rules partition the input space based on the BMU and the SEC. The solid lines, in the figure, indicate the original Voronoi regions that divide the plane based on the BMU. The dotted lines show how the original regions are subdivided to account for the SEC. The lines that define the subregions form the rule antecedent of the rules. Note that the entire input space is covered, with each of the subregions representing one rule.

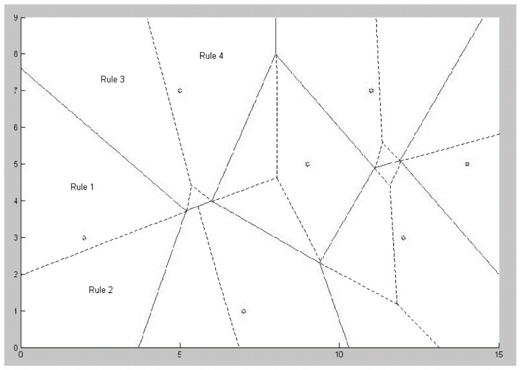

Figure 3 Deterministic Rule Coverage

These rules are specifically designed for the DCS structure implemented by the IFC project. The rule antecedent gives a set of constraints that geometrically define a region of the input space based on a specific BMU and SEC pair. The consequent of the rule then gives the DCS output based on this region.
The new rule format is

IF input $\in$ region 1 (input satisfies a set of constraints)
THEN output = multivariable linear expression

Below is an example of the new deterministic rules for a two-dimensional data set.

```
IF      (-4x + 2y >= -16) AND
        (-2x + 6y >= 12) AND
        (-6x + 0y >= -48) AND
        (7x + 2y >= 46.5) AND
        (2x + 4y >= 28) AND
        (-2x - 2y >= -32)

THEN    z = 0.2x - 0.1y + 1.7
```

Figure 4 shows how the rule antecedent partitions the plane for a two dimensional input domain. Each of the lines corresponds to one of the inequalities in the antecedent. All points in the shaded region below are associated with a BMU and SEC pair and therefore have the same output. If the domain is n-dimensional, then the n-dimensional space is partitioned by (n-1)-dimensional hyperplanes and the rules become much harder to read.

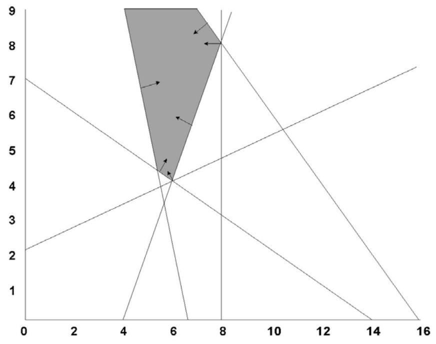

Figure 4 Rules Partitioning of the Plane

The algorithm for the deterministic rule extraction technique for the DCS is shown in Figure 5. Similar deterministic rule extraction techniques have been developed for feedforward neural networks (Setiono, Leow, and Zurada 2002).

Testing the New Technique

To test the accuracy of the deterministic rules extracted using the new algorithm, first the rules sets were generated. Then the rules were implemented and test data was used as input for both the neural network and the implemented rules. The two sets of output were compared for agreement. This was done on three different types of data.

The first test set was a hand-constructed two-dimensional DCS structure with seven centroids positioned as seen in Figure 2. The centroids were given a fully connected adjacency matrix and this description was translated into a DCS structure. Deterministic rules were extracted from this DCS structure. Twenty test cases composed of random data were generated, some with as many as 50,000 data points in them. For all test cases, the rules output had 100% agreement with the DCS output.

The next data set used for testing was the Iris data. This data set has four continuous variables as input and one categorical variable as output. The DCS was trained on 4/5 of the data set before applying rule extraction. The entire data set was then used to test the rule output against the DCS output. It should be noted that the DCS was not allowed to learn and grow during the testing portion, only during the training portion. This test was run several times using different data to train. Again 100% agreement was achieved between the rules and the actual DCS output.

```
Inputs:
W = Weight vectors of the DCS centroid of Voronoi region
A = Adjacency Matrix
Output:
Rules that geometrically describe a partition of the input
space with associated outputs
Procedure:
Let W define a Voronoi diagram that partitions the input
space.
Let A determine neighboring regions in the Voronoi
diagram to find BMU and SEC pairs.

For each $w \in W$ (centroid of Voronoi region and BMU)
        Calculate Voronoi region boundaries.
        For each $v \in W - \{w\}$ such that v is a neighbor
        of w
        (v is the centroid of neighboring region and SEC)
                -Determine boundaries that divide the
                region with centroid w into subregion
                based on v.
                -Determine antecedent based on
                boundaries defined by w and v.
                -Determine consequent equation
                determined by w (BMU) and v (SEC).
                Write rule.
```

Figure 5 Deterministic Rule Extraction Algorithm

The final test of the rule extraction algorithm was done on the flight data. This data was generated using a flight simulator (Perhinschi 2002) and the sets exhibited several different flight maneuvers. The data was used to train the neural network and then the rules were extracted. The data was applied to the fixed DCS and the rules with 100% agreement being achieved in this case as well.

Using Extracted Rules for V&V of the DCS

The following scenarios illustrate two examples of how extracted rules can be used in the V&V process. The neural network in the scenarios described below is a MATLAB implementation of the DCS neural network that ISR has developed for testing purposes. This neural network has the same characteristics as the DCS that was implemented in the IFC mentioned earlier.

Scenario 1: Human Understandable Rules Led to Identification of Coding Error

The original rule extraction algorithm, which generated human understandable rules, is based on how an input stimulus is matched to a centroid of the DCS or its BMU. The human understandable rules support verification inspection methods. Each input stimulus results in the selection of a BMU internal to the DCS network. The BMU is considered the centroid of a cell and each input that related to that BMU is considered to be a member of that cell. The human understandable rules were generated to describe each cell. The minimum and maximums of each input variable related to a specific BMU are determined to form the interval for the rule antecedents. The minimum and maximum of each output variable associated with this cell are used to form the interval for the rule consequent. Any BMU that did not have input stimulus matched to it did not generate a rule. (See rule algorithm Figure 1)

When the human understandable rule algorithm was applied to a DCS network that had been trained on the Iris data, a discrepancy was noted between the number of rules generated and the number of nodes within the DCS network. There were fewer rules than nodes. This implied that for the set of input data we used to train the neural network, a node was established that never matched any of the other data, and thus these BMUs did not have corresponding rules. Debugging and execution traces pointed to a problem in some of the DCS code that had been optimized to run within a MATLAB environment. The original IFC DCS code was developed within the C programming language. For optimization purposes, when the code was moved into a MATLAB script for experimentation, all usage of 'for' loops were removed and replaced with vectorized math. One of the lines of code used for the optimization searched for BMUs within an entire array at once, incorrectly omitting the use of an indicator of the current number of centroids and thus including entries within the BMU array that had not yet been used. At times, these undefined nodes were actually better at matching the input than any one of the existing nodes; therefore the DCS manipulated these non-assigned nodes when it should not have.

The result was that undefined nodes that had not been assigned were modified instead of the real nodes. These non-assigned nodes showed up as having non-zero values and appeared to be actual nodes upon visual inspection of the DCS structure, but did not actually exist in the structure of the neural network. DCS was losing some potential learning within these nodes. The rules ignored these non-assigned nodes since they were not able to become BMUs and that led to the discrepancy between the rule output and the neural network output.

When the line of code that controlled this was identified and modified to ignore non-assigned nodes, then DCS output more accurately matched the human understandable rules output. The rules gave insight that could not easily be captured by inspection of the neural network parameters or through testing.

Scenario 2: Deterministic Rules Led to Identification of Two Coding Errors

The deterministic rules are based on the theoretical structure of the DCS and capture the partitioning of the input space into Voronoi regions, and thus they are designed to have 100% agreement with the output of the DCS network. However, testing of some of the first sets of deterministic rules showed that there was a large disagreement between the rules and DCS.

The rules were re-structured so that the antecedents were broken into a rule pertaining to each BMU, and then under the BMU rules, each neighboring SEC rule was present. This allowed comparison to see if the errors between the rules and DCS were based upon BMU selection, SEC selection, or within the consequent.

By comparing the BMU chosen by the DCS with the specific rule that corresponded to the input, it was discovered that the BMU selection was consistent between the rules and the DCS. However, it was discovered that the selection of the SEC was not matching between the two. This then led to investigation of the recall function within the DCS code.

In the DCS recall function, two errors within the same line of code were discovered. One was related to substitution of the 'max' for 'min' commands within DCS. For the recall function to perform properly, the smallest Euclidean distance is always used to identify the closest node to a stimulus. This is true also when selecting the SEC from among a BMU's neighbors. But the code showed that the max function was being used in place of the min function. This would subsequently show up within the DCS recall function as the DCS always selected the BMU neighboring node furthest away from the stimulus.

Further, this same line of code contained an incorrect reference to the strengths of the BMU neighborhood connections rather than the distances of the neighbors from the stimulus. This was quite a significant error, but due to the robustness of the DCS network and the small values on which the network was learning, the mistake was masked much of the time. Normal testing of the DCS showed that it could achieve accuracies above 90%, even with this error present.

The line of code was changed to consider the distances rather than the connection strengths and to choose the min instead of max. The rules and DCS were compared again. This gave the expected results of 100% agreement. The deterministic rules were deemed a success because they had allowed for the discovery of two coding errors, which were not readily apparent during inspection of the structure or testing. It should be noted that these errors were only in the experimental optimized MATLAB version of the code and not in the implementation that was the IFC system.

Conclusion

The goal of this research was to demonstrate that rules extracted from a neural network could be used to assist in the V&V of the neural network in a safety- and mission-critical application. The rules are viewed as a descriptive representation of the neural network knowledge. This representation of the inner knowledge can be used to help the developer or V&V practitioner better understand whether the neural network is functioning as expected and assist in the process required to certify the neural network for high assurance systems.

A developer or V&V practitioner might use rule extraction for different purposes throughout the software development life cycle. There are areas of impact for rule extraction in concept, requirements, design, implementation, and testing activities. Rules can also be used to incorporate hazard and risk analysis knowledge into the neural network (Kurd 2003). Some of these uses have been outlined in a previous paper (Darrah, Taylor, and Skias 2004) and in *Methods and Procedure for the Independent Verification and Validation or Neural Networks* (Taylor et al. 2004). A comprehensive list of uses for rule extraction across the life cycle is included in the *Practitioner Guidance for the Independent Verification and Validation of Neural Networks* (Taylor, Darrah, and Pullum 2005), guidance that complements the IEEE 1012-1998 Standard for Verification and Validation (IEEE 1998).

Many tools and techniques have been created to extract rules from specific types of neural networks. The ultimate goal is to develop a practical, general rule extraction tool for V&V and other purposes. ProLogic, Inc. and ISR have developed the prototype for such a tool under a Phase I STTR project funded by NASA ARC. The tool, called NNRules, accepts as input a formal specification of the trained neural network and uses neural network rule extraction algorithms to translate the neural network into an equivalent set of rules. These rule-based systems, which represent the neural network's knowledge, have a visible and potentially human readable, decision logic that supports a robust set of verification techniques. NNRules embeds neural network rule extraction technology in a usable tool that will dramatically increase the ability to V&V high assurance neural network systems. The team will seek Phase II funding to continue development of a marketable tool.

Acknowledgements

This research was sponsored by NASA Goddard Research Center through the NASA Independent Verification & Validation Facility, under Research Grant NAG5-12069 and NASA Ames Research Center through a subcontract to ProLogic, Inc. under contract number NNA04AA20C. Special acknowledgments go to Dan McCaugherty.

References

Darrah, Marjorie, Brian Taylor and Spiro Skias. 2004. Rule Extraction From Dynamic Cell Structure Neural Network Used in a Safety Critical Application. *In Proceeding of Florida Artificial Intelligence Research Society Conference*, Miami FL, May 17-19, 2004.

Bruske, Jorg and Gerald Sommer. 1994. Dynamic Cell Structures. *In Proceedings of Neural Information Processing Systems (NIPS)*, 497-504.

Fritzke, B. 1994. Growing Cell-Structures – a Self-Organizing Network for Unsupervised and Supervised Learning, *Neural Networks*, 7(9): 1441-1460.

Martinetz, T. M. 1993. Competitive Hebbian Learning Rule Forms Perfectly Topology Preserving Maps. In *Proceedings of International Conference on Artificial Neural Networks* (ICANN) 427-434. Amsterdam: Springer.

Jorgensen, Charles C. 1997. Direct Adaptive Aircraft Control Using Dynamic Cell Structure Neural Networks. NASA Technical Memorandum 112198, NASA Ames Research Center.

Setiono R., W. K. Leow, and J. M. Zurada. 2002. Extraction of rules from artificial neural networks for nonlinear regression. *IEEE Transactions on Neural Networks* 13(3): 564-577

Fisher, A. 1936. *Annals of Eugenics* 7:179-188.

Perhinschi, M. G., G. Campa, M. R. Napolitano, M. Lando, L. Massotti, and M.L. Fravolini. 2002. A Simulation Tool for On-line Real-Time Parameter Identification. AIAA Guidance Navigation and Control Conference, Aug 8-10 Monterey, CA.

Kurd, Zeshan, and Tim Kelley. 2003. Safety Lifecycle for Developing Safety Critical Artificial Neural Networks. *In Proceedings of 22nd International Conference on Computer Safety, Reliability, and Security* (SAFECOMP'03) 23-26 September 2003.

Taylor, Brian J., Marjorie Darrah, Laura Pullum, James T. Smith, Leon Luxemburg, Spiro Skias, and Bojan Cukic. 2004. *Methods and Procedures for the Independent Verification and Validation of Neural Networks*. Technical Report prepared by Institute for Scientific Research, Inc. (ISR) for NASA Independent Verification and Validation Facility under grant NAG5-12069.

Taylor, Brian J., Marjorie Darrah, Laura Pullum, 2005. *Practitioner Guidance for the Independent Verification and Validation of Neural Networks*. Technical Report prepared by Institute for Scientific Research, Inc. (ISR) for NASA Independent Verification and Validation Facility under grant NAG5-12069. Forthcoming

Institute of Electrical and Electronics Engineering, Inc (IEEE), Software Engineering Standards Committee. 1998. *IEEE Standard for Software Verification and Validation (IEEE 1012-1998).* New York, NY.