

# Integrating A Flexible Modeling Framework (FMF) with the Network Security Assessment Instrument to Reduce Software Security Risk

David P. Gilliam and John D. Powell  
*Jet Propulsion Laboratory, California Institute of Technology*  
*David.Gilliam@jpl.nasa.gov, John.Powell@jpl.nasa.gov*

## Abstract

The network security assessment instrument is a comprehensive set of tools that can be used individually or collectively to ensure the security of network aware software applications and systems. Using the various tools collectively provide a distinct advantage for assuring the security of software and systems. Each tool's resulting output provides feedback into the other tools. Thus, more comprehensive assessment results are attained through the leverage each tool provides to the other when they are employed in concert.

Previous portions of this work were presented at the IEEE Wet Ice 2000 and 2001 Workshops and are printed in those proceedings.

This paper presents a portion of an overall research project on the generation of the network security assessment instrument to aid developers in assessing and assuring the security of software in the development and maintenance lifecycles. This portion, the Flexible Modeling Framework (FMF), focuses on modeling requirements and early lifecycle designs to discover vulnerabilities that result from interaction between system components that are either under development in a new system or proposed as additions to an existing system. There are early indications that this new approach, the Flexible Modeling Framework (FMF), has promise in the areas of network security as well as other critical areas such as system safety. Information about the overall research effort regarding network security is available at <http://security.jpl.nasa.gov/rssr>.

## 1. Introduction

The National Aeronautics and Space Administration (NASA) has tens of thousands of networked computer systems and applications. Software Security is a major concern due to the risk to both controlled and non-controlled systems from potential lost or corrupted data, theft of information, and unavailability of systems,

especially mission critical systems. The cost to NASA if mission critical systems were compromised, would be enormous if these systems were brought down or erroneous data sent to a spacecraft. This research examines formal verification of IT security of network aware software and systems through the creation of a security assessment instrument for the software development and maintenance life cycle. [1,2,3,4,5,6] The network security assessment instrument is composed of 5 parts:

1. A Vulnerability Matrix
2. Additional Security Assessment Tools (SATs)
3. A Property Based Testing (PBT) Tool, and
4. A Flexible Modeling Framework (FMF)
5. A Software Security Checklist (SSC)

The vulnerability contains vulnerability descriptions and the code used to exploit them. The SATs are a collection of tools available to test for potential weaknesses of software code. The PBT tool performs formal verification of properties at the code level. Like the PBT tool, the FMF formally verifies properties over the system. However, the FMF performs this action at the abstract level when code may or may not yet exist. The SSC will provide software code developers with another instrument for writing secure code for network aware applications. An ongoing effort is underway with the Multi-Mission Encryption Communication System (MECS) to pilot the usage of this security assessment instrument.

## 2. The Flexible Modeling Framework (FMF)

An innovative verification approach, which employs model checking as its core technology, is offered as a means to bring software security issues under formal control early in the life cycle. [1,2,3,4,7,8,9] The Flexible Modeling Framework (FMF) seeks to address the problem of formal verification of larger systems by a divide and conquer approach. First verifying a property over portions of the system, then incrementally inferring the results over

larger subsets of the entire system. As such the FMF is a: 1) system for building models in a component based manner to cope with system evolution over time and, 2) an approach of compositional verification to delay the effects of state space explosion. Thus allowing property verification results of large and complex models to be examined and extrapolated appropriately.

Modeling in a component-based manner involves the building of a series of small models, which will later be strategically combined for system verification purposes. This correlates the modeling function with modern software engineering and architecture practices whereby a system is divided into major parts, and subsequently into smaller detailed parts, and then integrated to build up a software system. An initial series of simple components can be built when few operational specifics are known about the system. However, these components can be combined and verified for consistency with properties of interest such as software security properties.

The approach of compositional verification used in the FMF seeks to verify properties over individual model components and then over strategic combinations of these components. The goals of this approach are to: 1) infer verification results over systems that are otherwise too large and complex for model checking from results of strategic subsets combinations while minimizing false reports of defects; 2) retain verification results from individual components and component combinations to increase the efficiency of subsequent verification attempts in light of modifications to a component.

### 3. FMF Verification

The FMF approach's verification strategy for systems, whose entire model is too large for MC, is to first verify the property in question (p) over each individual component individually. Next, p is verified over unique component combinations of 2,3,4... that are built up until no unique component combinations, whose state space may be model checked remain. During this combination and verification process, the relationships between the combinations are preserved such that for two arbitrary combinations x and y where x is a subset of y and the cardinality of x equals the cardinality of y minus 1, x will be considered the child of y in a tree structure of model checked model combinations. The result of maintaining these relationships is the generation of a tree of verified model component combinations (MCCs) with multiple root nodes. This tree is referred to as the Model Component Combination Tree (MCCT) The tree's leaf nodes consist of single verified components, the parents of leaf nodes consist of a combination of 2 components and

their parents consist of a combination of 3 components etc...

At some threshold, determined by the amount of available memory present on the verification-computing platform, the state space of MCCs becomes too large for state of the art MC. The combinations beyond this threshold may be systematically computed but not model checked and is referred to as the implicit portion of the MCCT. The portion of the MCCs that can be verified via MC is referred to as the explicit or verified portion of the MCCT.

A root node in the verified portion of the MCCT is implicitly connected to parents that represent MCCs whose state space size prohibits direct model checking due to memory constraints. The only exception to this assumption is when the state space of the entire system model is small enough to be model checked on the platform in a traditional way. However, the FMF approach offers benefits to modeling beyond MC state space explosion and can handle verification of this case as a trivial case where there will exist only one root node which represents the entire system and produces the traditional MC result over the entire system.

The implicit parent of an explicit root node follows the same relationship rules that explicit nodes follow with regard to makeup and cardinality. (See this section above) An explicit "root" node is a subset of its implicit parent and contains exactly one less model component in its combination. Additionally each parent (implicit and explicit) has a number of children equal to the number of components in its combination. Thus, the entire implicit portion of the MCCT can be systematically generated and probabilistic verification statements made. (See Following Sections)

In addition to supporting the FMF's compositional verification approach the MCCT allows for partial re-verification in light of system model changes. As the system evolves, related model components are updated to reflect changes. The modular style of modeling in components results in a localization of the effects of updating a model component. Since the relationships between components and MCCs and past verification results of model components and MCCs are maintained, only the modified component and MCCs in which it participates need be re-verified. This represents a significant savings in terms of computational efficiency during subsequent (re-) verification executions.

#### 3.1. Propagation of Verification Results

Each node (MCC) in the MCCT is a vehicle for retention of knowledge pertaining to the verification of the

MCC represented therein. Each node carries two primary pieces of knowledge directly pertaining to verification.

- A verification value ranging from 0 to 2 that describes whether the property in question holds over the associated MCC.
- A confidence rating ranges from 0 to 1 that describes the probability that the verification value produced is correct.

### 3.1.1. Verification Values

A verification value of 0 indicates that the property decidedly does not hold. In terms of network security properties for software, a property violation represents the discovery of a network security vulnerability. The indication as the verification value progresses from 0 towards 1 is that the predictability of the property not holding is diminishing towards undecidability and is completely undecidable at a value of one. As the verification value moves from 1 to 2 the predictability that the property holds is increasing with a maximal predisposition that the property holds over the MCC when the verification value is 2. The verification value will always be 0 or 2 in the explicit or verified portion of the MCCT because the MCCs are model checkable and thus a definitive verification result is obtained for that MCC. As verification values are derived for MCCs or nodes in the implicit portion of the MCCT the degree to which a property is believed to hold /not-hold over a given MCC begins to vary between 0 and 2 because no direct MC results are available. It bears noting here that the verification value in no way expresses confidence in the verification result expressed by the verification value.

Thus, a Verification Value of 1 means that information across the various model components in a MCC is in conflict. A separate confidence rating is used to express the degree of confidence that the heuristics have correctly identified conflicts. (See Section 3.1.2) Verification Values in the implicit portion of the MCCT are calculated by averaging the Verification values of its children. (See Figure 1) In the explicit portion the MCCT, the verification values of a node is not heuristically derived from its children because the application of MC to the parent's MCC is a definitive verification answer. As can be seen in Figure 1 it is possible to systematically propagate verification values over the full set of model components and thus the entire system model.

Verification values assist in determining early lifecycle repairs and assurances for a system before software security vulnerabilities propagate and become exponentially more expensive to repair. In relation to the MCCT these activities are guided by trying to maximize and/or preserve Verification Values across the MCCT. For example, the small system depicted in Figure 1 indicates that model component A ( $VV_A$ ) violates the property but the violation is mitigated individually by both model components B ( $VV_B$ ) and C ( $VV_C$ ). However, B ( $VV_B$ ) and C ( $VV_C$ ) taken together without benefit of any other model component violate the property. Further, even though together they violate the property in question in combination, individually each model component (B and C) satisfies it as well as correcting the property's violation in A ( $VV_A$ ). Consider that this set of component represents the system at the early design or architecture stage of the development life cycle. Sever important questions can be prioritized and addressed. First, what interaction between the behaviors represented

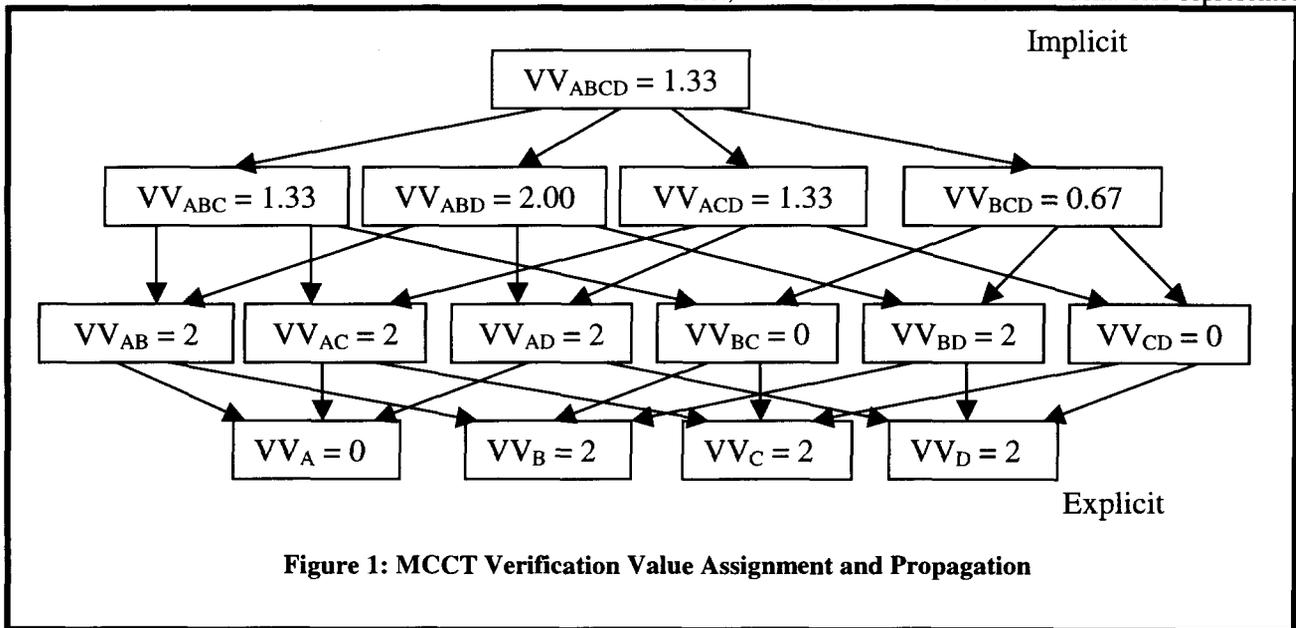


Figure 1: MCCT Verification Value Assignment and Propagation

by model components B and C causes the property to be violated? Identification of this anomaly guides the network security profession or software developer toward the root cause of the vulnerability. Secondly, does components B and/or C mitigate the property violation found in component A because they are explicitly required to do so or is it coincidental? The documentation must either reflect this reliance on B and C in the form of new / existing requirements or action must be taken to isolate and repair the transient vulnerability.

Similar steps may be taken to address the verification value of 0 in MCC  $VV_{CD}$ . It becomes apparent from this very small example that verification values in the FMF approach generates numerous interrelated questions. However, this is considered a strength of the FMF approach because the questions and issues are brought out very early in the lifecycle for consideration. Further, as early decisions are made and captured, efficient localized updates of the system model are supported through the modular nature of the model components in a very agile manner. Then, issues affected by subsequent changes are automatically revisited though the required localized re-verification of affected combinations. This last feature of the approach is considered a failsafe and not a substitute for good practices such as documentation of decisions and emergent requirements.

### 3.1.2. Confidence Ratings

Confidence ratings for MCCs reflect the degree to which the heuristics of the FMF approach believe that the corresponding verification value correctly decides the result of the property verification over the MCC. It bears noting here that the confidence rating does not serve to improve decideability. Rather it serves to project the confidence in the decision once it is made. For example, when the verification value is at or very near 1 a high confidence rating means that the approach is very sure it the verification answer can't be derived from the information available. Conversely, a low confidence rating when the verification value is (near) 1 means that insufficient information to decide the verification is available to decide the verification answer but existing information conflicts to a large degree. Reasoning about confidence ratings over verification values provides useful distinctions that may be used to guide future action. (See Table 1)

## 4. Non-Uniform Component Resolution

There are numerous instances in which one must view a system or set of systems at a very abstract level before

Conf. Rating	Verif. Value	Description
1.0	2.0	Highest confidence of No Property Violation
0.99-0.7	2.0	Reasonably High Confidence of No Property Violation
0.69-0.00	2.0	Questionable to no confidence of No Property Violation
1.0	0.0	Highest confidence of Property Violation
0.99-0.7	0.0	Reasonably high confidence of Property Violation
0.69-0.00	0.0	Questionable to no confidence of Property Violation
0.99-0.7	1.0	Reasonably High confidence Property (Non-) Violation cannot be Predicted
0.69-0.0	1.0	Questionable to Low confidence Property (Non-) Violation cannot be Predicted. Non-Uniform Component Resolution may yield productive predictions

**Table 1: Conf. Rating and Verif. Value Descriptions**

examining one or more parts in greater detail. In the security arena one will view a large network system from an extremely high level where protocols must be understood and systems within it are arbitrary connected entities. When building a system that will interact with a network, the focus on that particular system entity becomes more detailed but the remainder of the network is still viewed very abstractly. The level of detail in which a component/entity is modeled and examined is referred to as its resolution throughout this paper. As specific interactions with other systems are defined the resolution of those systems necessarily becomes somewhat more detailed to deal with property verification issues.

A system under development will be decomposed at a high level into various network aware and non-network aware components. Subsequently, components will be viewed with varying levels of resolution. Consider a system where network aware components such as OS and network functionality are interacting with non-network aware components such as routines for a printer. At this level there are network aware components on the computer system, components making up the network environment, and components making up the local printer routine.

In the FMF approach, components are viewed not only as a matrix of combinable components created at a given level of detail (resolution), but also as a 3-dimensional space of models whereby component versions of different resolutions may be selected to manage state space explosion. Therefore, the tolerable state space may be spread across several components in varying amounts,

which together from a state space that is feasible for MC. The approach of building model components, as opposed to a single model, allows localized modification and enhancement of model behavior and detail in order to examine subsets of the system at various non-uniform levels of resolution.

The FMF component methodology provides the ability to make tradeoffs in resolution between components while maintaining the size of the state space within tolerable limits. The process of enhancing model components as more is learned about the system results in a series of component versions. When archived for later use, they provide a readily usable facility for producing component combinations with nonuniform resolutions. This is done by selecting the components for the combination and then specifying what version (resolution) of the component to use.

The ability to investigate one component in detail (Cn.3 level) is facilitated by accepting lower levels of resolution (Cn level) in other components. Raising the resolution in one component increases the state space. Conversely, lowering resolution decreases the state space. Therefore, to gain maximum benefits from the available memory resources the resolution is increased until the threshold is almost reached. The analyst may then continue to make tradeoffs to probe various parts of the system in greater detail by increasing resolution on one component and decreasing it on others.

It is important to note that in practice resolution levels in terms of contribution to state space size by a component is not necessarily uniform across components in a given level. The resolution levels only preserve order of size of a component. The generation of differing levels of resolution within a component is dictated by the circumstance of the natural process of updating model pieces as the system becomes more robust.

Consider an example where a local print routine with no network capability is locally connected to an OS that must also interact with the Internet. Upon initial examination the printer routine would not appear to have to concern itself with security issues. To gain a higher confidence than this is the case the components within each system (Network, OS, and Print Routine) must be examined at a higher resolution. Examination of all components at a high level of resolution may be an infeasible task for MC with reasonable memory constraints. Thus a process is undertaken to gradually increase the “resolution” on the OS and the print routine while lower the “resolution” on the network and print routine’s model components.

While printer components like CP1, do not directly interact with the network it may now be possible for them to interact with otherwise secure system components in a

manner that renders them unsecured. An attack similar to this scenario above has been seen before. The attack managed to exploit interactions between system components to send a message to the print routine with additional data embedded in it. At a higher level it appears that a component in the OS is sending a routine message to a non-networked printer – a harmless interaction. Thus, the message is not scrutinized by the OS because it is a non-network aware activity. Consequently, the message is allowed to proceed to a seemingly noncritical area (the non-networked printer). However, because the print routine did not address network security concerns, it failed to identify that its normal responses back to the computer system were actually giving an unauthorized outside party root access to the system. From there the attacker had access to the computer and potentially the entire network.

### 5. Other Verification Systems

In the network security arena, an integrated approach that includes the FMF as a model-based verification element, for assessing security vulnerabilities is being explored. The other parts of the Security Assessment Instrument are PBT and the VMatrix. PBT is an approach that allows the analyst to systematically test an implementation. The Vmatrix, examines vulnerabilities, exposures and the methods used to exploit them. Vulnerabilities and exposures are listed along with their Common Vulnerabilities and Exposures (CVE) listing. [2] The individual parts of the Security Assessment Instrument can be used separately or in combination (See Figure 2) to provide the additional benefits.

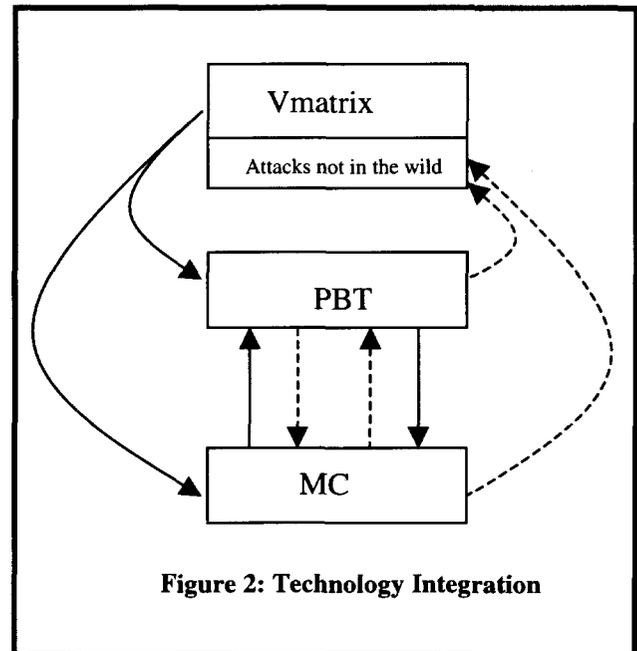


Figure 2: Technology Integration

## 6. Conclusion

Reducing the number of vulnerabilities in software systems is critical in computer systems that perform safety critical functions and/or make use of network connectivity. The use of formal approaches such as MC enhances the ability of developers and analysts to discover vulnerabilities arising through unsafe interactions between systems and/or otherwise safe software components. The FMF approach provides the benefits derived from MC while mitigating limitations posed by system size and complexity as well as requirements and design volatility during the early lifecycle phases. The FMF attempts to capitalize on the benefits of existing technologies in a manner that maximizes usability and minimizes duplication of effort between approaches.

Integrating software security and safety into existing and emerging practices is critical for developing high quality software. The Flexible Modeling Framework (FMF) offers a formal approach achieving such integration throughout the software development and maintenance life cycles. The approach seeks to maximize these benefits by attempting to integrate with, as opposed to replacing, existing verification technologies.

## 7. Acknowledgements

The research described in this paper is being carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration, and the University of California at Davis under a subcontract with the Jet Propulsion Laboratory, California Institute of Technology

## 8. References

- [1] D. Gilliam, J. Kelly, M. Bishop, "Reducing Software Security Risk Through an Integrated Approach," Proc. of the Ninth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (June, 2000), Gaithersburg, MD, pp.141-146.
- [2] G. Fink, M. Bishop, "Property Based Testing: A New Approach to Testing for Assurance," ACM SIGSOFT Software Engineering Notes 22(4) (July 1997).
- [3] M. Bishop, "Vulnerabilities Analysis," Proceedings of the Recent Advances in Intrusion Detection (Sep. 1999).
- [4] J. Dodson, "Specification and Classification of Generic Security Flaws for the Tester's Assistant Library," M.S. Thesis, Department of Computer Science, University of California at Davis, Davis CA (June 1996).
- [5] D. Gilliam, J. Kelly, J. Powell, M. Bishop, "Development of a Software Security Assessment Instrument to Reduce Software Security Risk" Proc. of the Tenth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Boston, MA, pp 144-149.
- [6] D. Gilliam, J. Powell, J. Kelly, M. Bishop, "Reducing Software Security Risk Through an Integrated Approach", IEEE Goddard 26th Annual Software Engineering Workshop
- [7] W. Wen and F Mizoguchi. Model checking Security Protocols: A Case Study Using SPIN, IMC Technical Report, November, 1998.
- [8] G. Holzmann. Design and Validation of Computer Protocols. Prentice Hall 1990; ISBN: 0135399254 .
- [9] J. R. Callahan, S. M. Easterbrook and T. L. Montgomery, "Generating Test Oracles via Model Checking," NASA/WVU Software Research Lab, Fairmont, WV, Technical Report # NASA-IVV-98-015, 1998.