Better V&V for Critical Flight Systems

Xuegao An* and Herbert Hecht†
SoHaR Incorporated, Culver City, CA 90230

Critical aircraft systems are becoming more dependent on software. This brings with it the need to establish that the software will perform safely and reliably through all flight regimes, including emergencies. Verification and validation (V&V) is a key process for meeting that need for both military and civil procurements. For software developed with the UML (Unified Modeling Language), we are describing MOVAT, a computer-aided approach that is much more disciplined and repeatable than current practice and at the same time offers a considerable reduction in labor and schedule. MOVAT generates a Failure Modes and Effects Analysis (FMEA) directly from UML artifacts (use case diagrams during the requirements phase and class diagrams later on), and a Timed Petri Net (TPN) analysis of timing related problems from collaboration diagrams. These identify areas of greatest failure potential (expressed in severity categories) as well as associated detection capabilities and compensation (recovery) mechanisms. While software FMEA has been described and used previously it has generally been based on "functions", a subjective concept. MOVAT uses operations of classes, clearly documented software constructs. When all operations in a class have been analyzed we can claim that the class has been completely evaluated, equivalent to using a parts list to establish that a hardware FMEA is complete. The FMEA and TPN permit V&V to concentrate on the software constructs most critical to safety of flight and to evaluate coverage of detection and recovery mechanisms. The emphasis is in most cases shifted from assessment of the functional software to assessment of the detection and recovery segments. These are usually much simpler and more standardized that the software elements that they protect and therefore the cost of V&V will be reduced. The procedure will be demonstrated on an autonomous active/standby redundancy management system, a design element encountered in fuel management, pressurization, and communication systems but also applicable on a grander scale to the leader/follower role assignment of a swarm of UAVs. The examples will describe applications during the requirements and coding phases. The research reported on here has been sponsored by the DARPA MoBIES project and AFRL.

Acronyms

AIAA:	American Institute of Aeronautics and Astronautics
AFRL:	Air Force Research Laboratory
CMM:	Capability Maturity Model
DARPA:	Defense Advanced Research Projects Agency
FMEA:	Failure Modes and Effects Analysis
MoBIES:	Model-Based Integration of Embedded Software
MOVAT:	Model-Based Verification and Assurance Tool
PHA:	Preliminary Hazards Analysis
TPN:	Timed Petri Net
V&V:	Verification and Validation
UAV:	Unmanned Aerial Vehicle
UML:	Unified Modeling Language

* Senior Research Engineer, SoHaR Inc., 5731 W. Slauson Ave. Suite 175, Culver City, CA 90230, Member of AIAA.
† Chief Engineer, SoHaR Inc., 5731 W. Slauson Ave. Suite 175, Culver City, CA 90230, Senior Member of AIAA.

I. Introduction

Critical aircraft systems are becoming more and more dependent on software. In order to make sure the system will perform safely and reliably through all flight regimes, including emergencies, the software V&V has to effectively deal with the increased complexity and functionality. Although industry standards and government documents provide some guidance on the conduct of V&V, most observers will agree that there is considerable variability in V&V. The process frequently becomes open-ended and very costly.

The MOVAT approach described here automates many steps of V&V, rationalizes others, and provides a completion criterion. The automation is primarily aimed at programs that are generated from the UML models; other features are more broadly applicable.

MOVAT extracts *classes* and their *operations* from the UML class diagrams, puts these into hierarchical order, and prompts the analyst to assign to each *operation* one or more potential failure modes from a pull-down menu. Subsequent screens propagate the effects of these failure modes to the system level where a limited number of failure effects and standardized severity rankings are specified in consultation with the system engineers. For all failure modes, but particularly for those with the highest severity rankings, the analyst specifies means of failure detection and recovery. All of this data are saved into a database and then presented in FMEA worksheets.

Potential timing related failures are made explicit by means of a TPN presentation that can be constructed from the UML collaboration diagrams. Failure modes detected by the TPN are then incorporated into the FMEA.

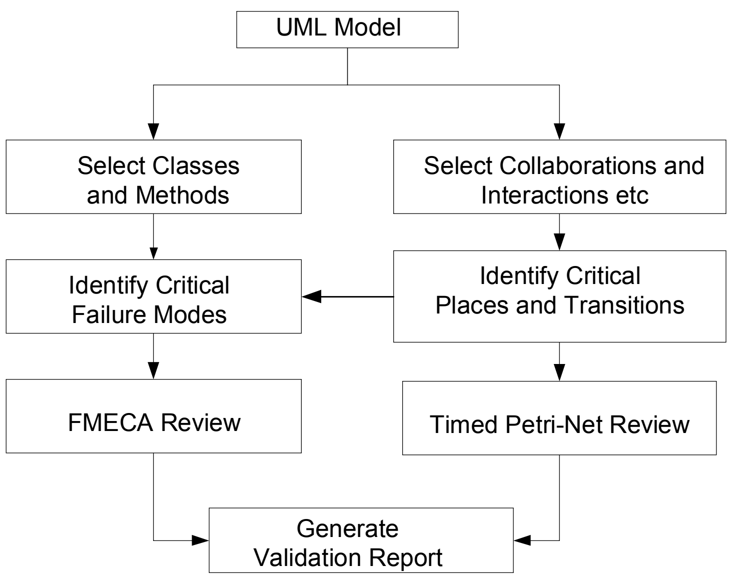

Figure 1. MOVAT Structure

The processes that accomplish this are shown in Figure 1. The left branch of the figure corresponds the generation of the FMEA, which will be described in Section III later, and the right branch corresponds to the generation of the TPN, which will be described in section IV.

The FMEA format provides a systematic way of presenting potential failure modes of a program and defenses against them (detection and recovery). When all *operations* of a UML-based program have been listed in the FMEA worksheet, the coverage is just as complete as when all parts have been listed in a hardware FMEA. This measure of completeness cannot be achieved by any other V&V approach in the public domain. The automated capture from the UML class diagrams and collaboration diagrams reduces both labor and the potential for errors and omissions.

The reviews are facilitated by the severity rankings of the FMEA. Instead of having to analyze the correctness of each element of the program, the reviewer can concentrate on the effectiveness of the protective measures, detection and compensation mechanisms. Because detection and compensation (recovery) procedures are much more standardized that functional program elements, the review process is speeded up and made more transparent. The capture of every operation used in the program assures completion of the process.

The principles of this approach can be applied to non-UML programs by structuring these in *use case* format. The *use case* format is also valuable for preliminary V&V activities in the UML based development before the program structure is complete.

In section II, we will first describe the V &V for pre-coding phase about software requirements, which often are documented in *use case* diagrams.

II. V&V for Pre-Coding Phase

The emphasis for V&V activities prior to coding is to prevent failures that could result in unsafe conditions or that could prevent completion of the mission. Early in the software requirements phase, any software component

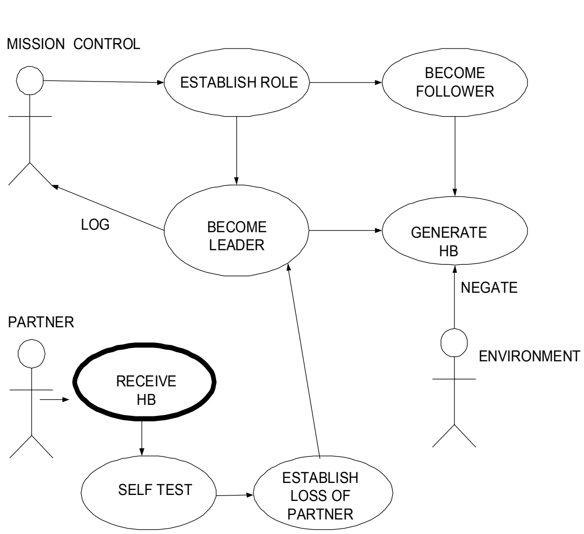

Figure 2. Use case diagram

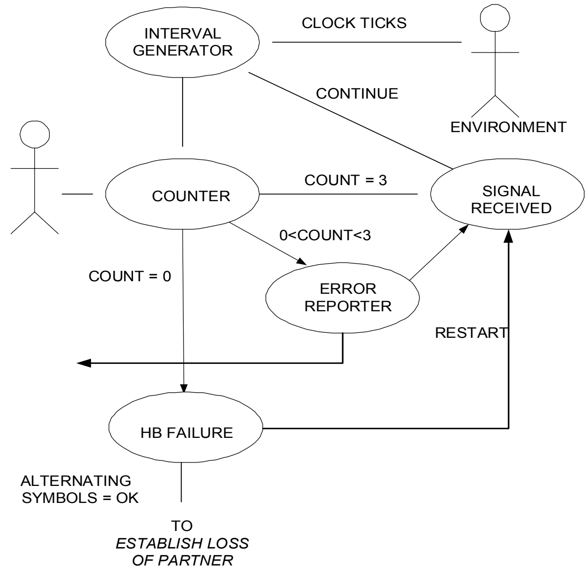

Figure 3. Details of receive HB

must be assumed to be prone to failure, and therefore review and analytical efforts are aimed at mitigating the effects of failures, particularly of those that impact safety and mission success. A good starting point for these activities is a document listing potential hazards such as a PHA that should have been generated in the Requirements phase.

As the structure of the software evolves, *use case diagrams* can be generated automatically or manually. Figure 2 shows an example of a use case diagram for autonomous assignment of "leader" and "follower" roles for a swarm of two UAVs. The stick figures are called "actors" but do not necessarily represent persons. In our example *Mission Control* may be staffed or automated. The other two figures are computer outputs. The ovals in the figure denote software modules and the arcs show potential propagation of failure effects. With some changes in nomenclature the same use case diagram can represent activation of the standby element when the active one fails in any system that employs standby redundancy. It therefore depicts a frequently encountered structure. A switch in roles is initiated when the standby or follower unit fails to receive a heartbeat (HB) from the active or leader unit. The key function for initiating the switch-over is the *Receive* HB module, shown in heavy outline in the figure. A review of failure modes and effects for this module is shown in Table 1.

A failure in which a received heartbeat is not recognized is termed a passive failure. An active failure is one in which a heartbeat is recognized although it has not been received. Under routine conditions (no other failures) active failures have no effect and thus they are not detected unless special testing is provided. When the partner fails (does not send heartbeats), if there is an active failure in the *Receive HB* module, the loss of partner will not be recognized. The follower will miss becoming the leader, thus defeating the redundancy scheme and causing mission failure.

Failure Mode	**Pre-Failure Role**	
	Leader	Follower
Passive	Change Tactics	Become Leader
Active – Routine	No Effect	No Effect
Active – Partner Fails	Miss Change Tactics	**Miss Becoming Leader**

Table 1. Failure Modes and Effects for *Receive HB* Module

The early stage analysis performed here permits design changes to overcome undesirable failure modes with comparatively little project impact. An example of such a change is shown in Figure 3, an expanded representation of the *Receive HB* module. The design feature introduced to greatly reduce the probability of an active failure is to require receipt of three equally spaced pulses during a given interval to signify a valid heartbeat. Only when the counter reports exactly three heartbeats is a valid signal recognized. When zero heartbeats are received during an interval this constitutes a heartbeat failure. This condition is signaled to the *Establish Loss of Partner* module (see Figure 2) by absence of an alternating 0101 string. The alternating string is used to prevent a spurious high state on the line from being recognized as "ok". An output of the counter other than zero or three is evidence of a failure in either the transmission link or the software and results in an error report being sent to mission control. This provides early warning of a previously undetectable condition that can impair mission success.

The corrective measures taken on the critical software elements identified at this early development stage must remain under review. In particular, subsequent V&V steps must assure that the coding fully implements the functions that have been adopted in preceding steps, and test must provide assurance that the protective measures work as intended.

III. FMEA Generation from Class Diagrams

In a UML model of a software system, c*lass diagrams* represent the structure of the software and list all of the *operations* utilized by each class. Figure 4 shows the UML interface tool to retrieve classes and their methods, along with their association relationships. Information about collaboration diagrams or other timing related UML constructs can also be retrieved. They will be used to generate TPN as we will describe in section IV.

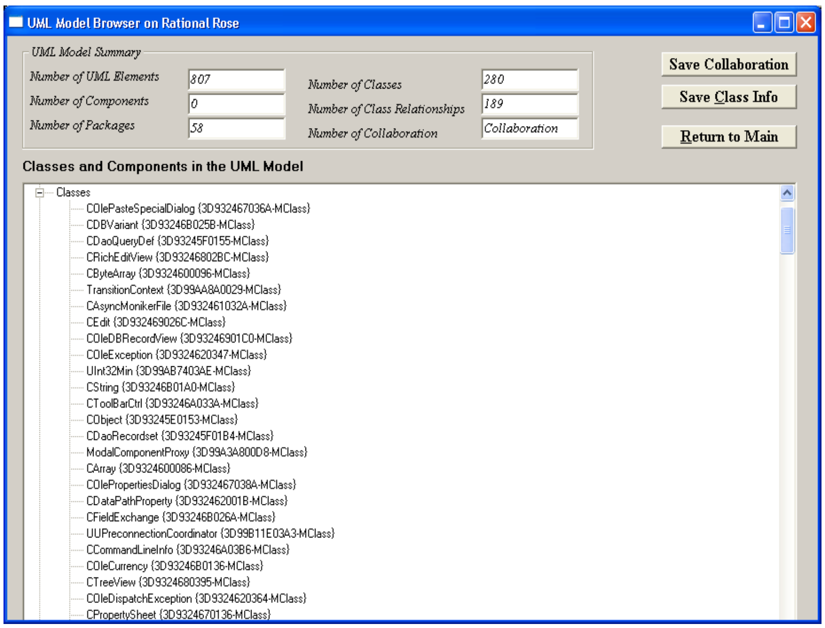

Figure 4. UML interface tool

The listing of the operations of classes represents the entry point for FMEA generation, and because all operations are listed in the diagrams the software FMEA generated in this manner will be as complete as a hardware FMEA generated from a bill of materials. The FMEA permits systematic evaluation of the failure modes and their effects for all of the operations. The system level severity of the effects is characterized by standardized scales. For military use, there are four categories (I to IV) with I stands for highest severity. There are 10 severity ranks (1 to 10) for automotive applications with 10 stands for highest severity. The extent of the failure prevention activities (including V&V) should be governed by the severity classification. Thus the objective of the FMEA generation is to provide the organizing scaffolding for V&V activities, such as reviews and testing, during the late development phases.

The activities to perform software FMEA include the following

a. Collection of class and operation data from the UML tool
b. Assignment of failure modes for each operation
c. Propagation of failure modes to failure effects and classification of the severity of the effects
d. Evaluation of protective measures (detection and compensation)
e. Review of the FMEA for completeness and for adequacy of detection and compensation

A screen shot of the implementation of steps a. and b. is shown in Figure 5. The top two rows are provided by the UML listing; the left part contains the nomenclature and the right part the description of the selected component. The next row is devoted to failure mode assignment, with the left panel containing a large listing of potential failure modes from which MOVAT selects and lists in the right panel those applicable to the highlighted operation based on key words in the description. The right arrow key permits the analyst to add a mode from the general list to the selection while the left arrow key removes a failure mode from the selection.

The next lower panel and the associated tabs permit the analyst to select the operating phase for which effects and mitigation will be specified. For avionics applications typical phases are taxi, take-off, cruise, approach, and maintenance. The effect of a given failure mode will vary, depending on the phase in which it occurs. A failure in the radar altimeter program will have no significant effect in the taxi and maintenance phases, minor effects in take-off and cruise, and possibly catastrophic effects during approach.

The identification of failure effects and of mitigation (items c. and d. above) is shown in Figure 6. The top part of the screen is identical with Figure 5 but the lower part shows the response to the Effect and Mitigation tabs.

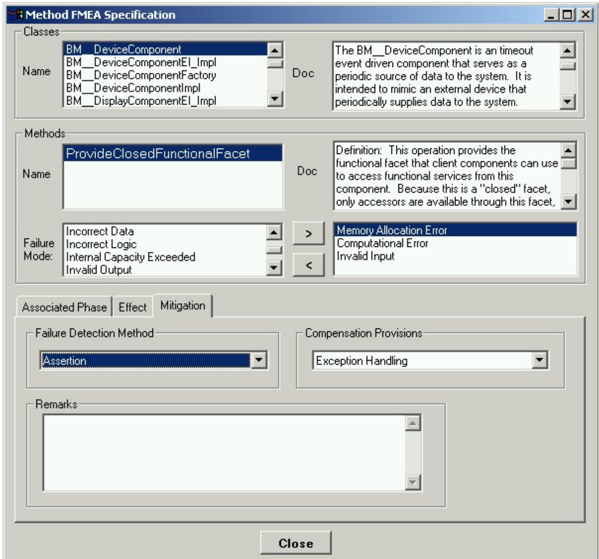

Figure 5. Assignment of failure modes

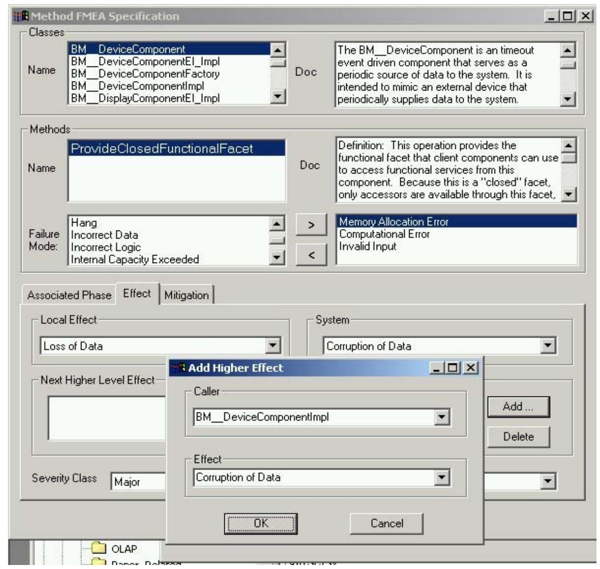

Figure 6. Selection of effects and mitigation

The effects of a given failure mode are typically evaluated at three levels: Local, Next Higher, and System, and this is the MOVAT default. For simple programs only a single effects level may be necessary while for an unusually complex one more than three levels may be required. The analyst can add or reduce effect levels from the MOVAT default. The system level effects are usually found in a system hazard analysis and are independent of the software analysis activities. For an avionics system typical system level effects are loss of aircraft, injury to passengers or crew, need for emergency landing, inability to reach destination, and unscheduled maintenance required. A severity category is associated with each system level effect, and again this decision is normally made by system engineering. Loss of a manned aircraft is a catastrophic event (Severity I), injury and emergency landing are major events (Severity II), mission impairment is a marginal event (Severity III) and unscheduled maintenance is a minor event (Severity IV).

The protective measures, sometimes called mitigation, should correspond to the severity of the effect. In most cases protective measures consist of a detection mechanism and a recovery or compensation feature. Typical detection mechanisms are assertions, time-outs, and comparisons and these usually are implemented as operations in UML and they can be tagged as detection operations. This facilitates (a) semi-automated association with the operations that are being protected, and (b) protecting the operations against modification or deletion without careful review of the effects on the detection capabilities. Typical compensation features include re-try, re-start, use of default values and invocation of an alternate routine for accomplishing the impaired function. Again, the operations that implement these features can be tagged and thus the recognition of compensation for the covered failure effects can be automated.

IV. Timed Petri Net (TPN) Generation

System interaction problems are not necessarily apparent from the FMEA constructed in this manner. Hardware interaction problems are likewise not usually apparent from an FMEA but experience over many decades has shown that failure effects due to interaction problems are usually the same as those caused by parts problems. If the system tolerates a signal outage due to a parts failure it will also tolerate the outage if it is due to an interaction problem. However, software is particularly sensitive to timing variations. Therefore a separate investigation, based on TPN, is described in this section.

Because there is no assurance that the analyst will recognize timing related failure modes, the FMEA generation will be augmented by generating TPNs. Potential timing conflicts become obvious in a TPN and the expected frequency of these problems as well as the effectiveness of solutions can be explored analytically or by simulation. The existence of timing related failure modes and potential protective measures are then entered into the FMEA and are utilized in V&V in the same way as other failure modes.

The essential activities to perform TPN analysis are:

a. Selection of collaboration diagrams from the software UML model

b. Transformation from the UML construct into TPN model
c. Analytical analysis or simulation of the TPN model

An example of a collaboration diagram is shown in Figure 7, which represents the exchanges required to generate a lock. Passage of time is denoted by arrows and usually runs down and to the right. An arc above a function indicates data generated and consumed within the same function.

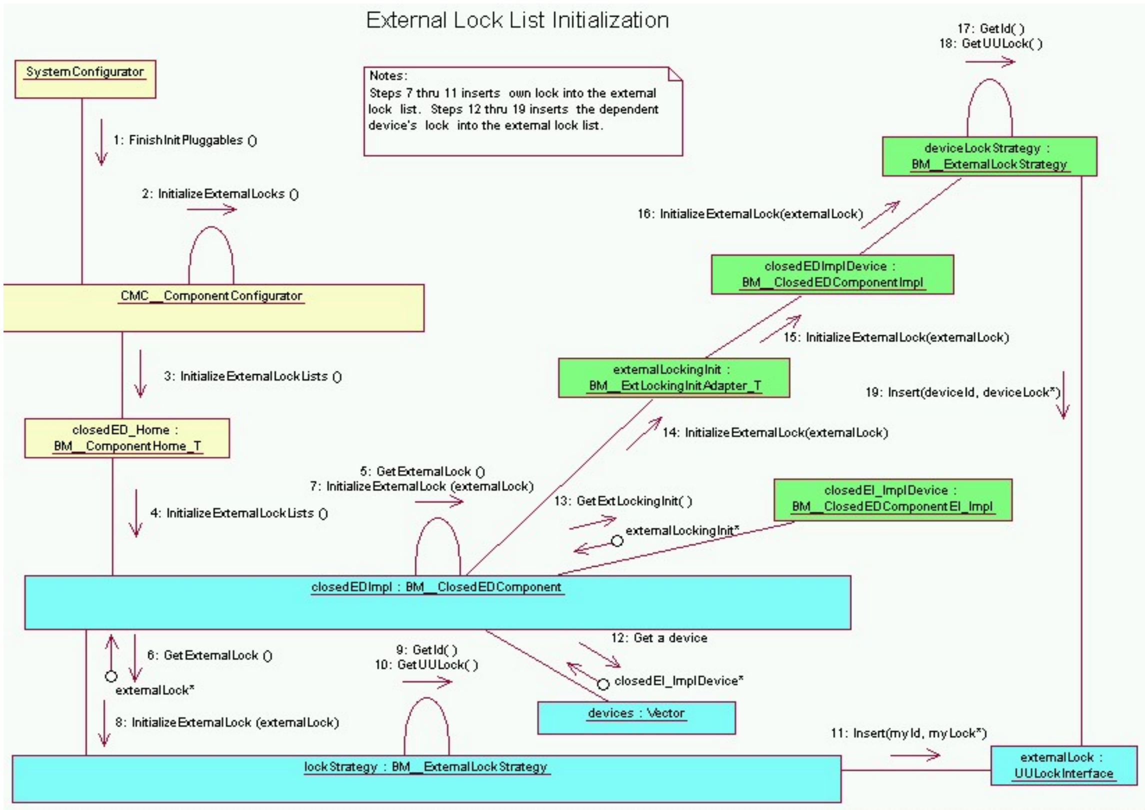

Figure 7. A sample collaboration diagram

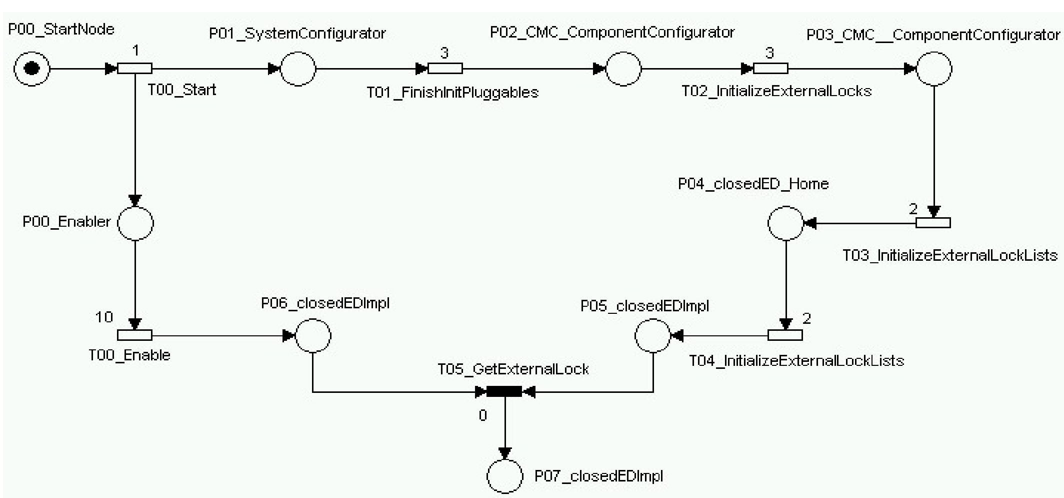

Figure 8. High level TPN model

The information in the collaboration diagram is retrieved using the UML interface tool mentioned at the beginning of section III. Such information regarding the sequence in which classes and operations interchange messages are then used to create TPN models. As part of the transformation, UML classes and operations become TPN places, and UML arrows become TPN transitions.

For the collaboration diagram shown in Figure 7, a high-level TPN model is shown in Figure 8, which permits immediate recognition of a timing problem at transition TO5. Open rectangles are timed transitions, with the average delay (in computer cycles) indicated by numerals. Black rectangles are immediate transitions.

To get the external lock it is necessary that both inputs to TO5 be available within the same computer cycle. While the average delay is 10 computer cycles in either path leading to TO5, random events may delay one or the other signal and thus cause inability to complete the task. Given the distribution function for the delays, the probability of a timing problem can be calculated or it can be obtained by simulation (the TPN can be executed). Any remedial measures (such as holding the individual lock permissions for one or more computer cycles) or protective measures, such a repeating the operation if it does not complete, can be evaluated in this manner.

V. Conclusion

A major benefit is that the FMEA is based on a file of *operations* generated and maintained by the UML software tool that develops the software. Thus the analyst is relieved of the need to partition routines into "functions". Since the operations are obtained from a complete file, it is possible to establish a completion criterion for the FMEA and also for the entire V&V. The generation of the FMEA is largely computer aided, with the analyst being prompted to select from context sensitive menus. Because the FMEA files reside with the program files, changes to the program will be immediately reflected in the FMEA file and the analyst will be prompted to accept automatically generated changes to the FMEA or to enter required changes. These features reduce the cost of FMEA generation, make it more objective and complete, and avoid the FMEA becoming "stale".

Because timing problems are not obvious in the MOVAT FMEA generation described in section III, it is desirable to supplement that procedure by using timed Petri Nets. The graphics as well as the executable version of the TPN can be created in a largely automated manner from the collaboration diagrams that is part of the UML file. Thus MOVAT includes a very powerful tool for detecting and correcting timing problems.

The combination of FMEA and TPN based on the UML model will enable better V&V of computer programs for critical applications. Better here means:

- Computer-aided, if not fully automated.
- Organic part of the software development and therefore updated as it changes.
- More complete, with achievable completion criterion.
- More objective, yet allow "expert" review.

Acknowledgments

The authors want to thank the sponsors of this research: DARPA as part of the MoBIES project headed by Dr. John Bay, and AFRL/VA under contract F33615-02-C-3253 with Raymond Bortner as the Technical Representative.

References

Goddard, P. L., "Software FMEA techniques," *2000 Proceedings Annual Reliability and Maintainability Symposium*, 2000, pp. 118-123.

Ammar, H. H., Nikzadeh, T., and Dugan, J. B., "A methodology for risk assessment of functional specification of software systems using colored Petri nets". *Proceedings Fourth International Software Metrics Symposium*, 1997, pp. 108-117.

Becker, J. C., & Flick, G. "A practical approach to failure mode, effects and criticality analysis (FMECA) for computing systems," *High-Assurance Systems Engineering Workshop*, 1996, pp. 228-236.

Bowles, J. B., "The new SAE FMECA standard," *1998 Proceedings Annual Reliability and Maintainability Symposium,* 1998, pp. 48-53.

Luke, S. R., "Failure mode, effects and criticality analysis (FMECA) for software," *5th Fleet Maintenance Symposium, Virginia Beach, VA (USA), 24-25 Oct 1995,* pp. 731-735.