Toward Better V&V

Herbert Hecht and Xuegao An
SoHaR Incorporated
Culver City, California

A. EXECUTIVE SUMMARY

1. Scope

This paper addresses computer programs for safety and reliability critical applications. Process based approaches, such as CMM, must be supplemented for these critical programs by assessment of the specific product by V&V, a costly and frequently open-ended process (it stops when the money runs out). 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 generated with UML (Uniform Modeling Language) tools; other features are more broadly applicable.

2. Overview.

MOVAT extracts *cases* and *methods* from the UML Case Chart, puts these into hierarchical order, and prompts the analyst to assign to each *method* 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 is then presented in a Failure Modes and Effects Analysis (FMEA) worksheet such as the one shown in Table 1.

Table 1. Example of FMEA Worksheet

Class Name	Method Name	Failure Mode	Failure Effect			Sever-ity	Detect-ion	Compen sation	Remarks
			Local	Next Higher	System				
BM__Com ponent	BM__Comp onent	Fail to Initialize Variables	Data Error	Late Response	None	IV	Assertion	Repeat	Sev.II if repeat fails
BM__Abstr actCompo nentHome	CreateAndIn it	Fail to Initialize Variables	Data Error	Stop	Stop	III	External	None	
BM__Abstr actCompo nentHome	FindByPrima ryKey	Computati onal Error	Crash	None	None	IV	Assertion	Alternate Find	Sev. II if alternate fails
BM__Abstr actCompo nentPieces	GetEquivale ntInterface	Faulty Message	Hang	Late Response	Temporary degraded response	IV	Assertion	Repeat	Use default if repeat hangs

Potential timing related failures are made explicit by means of a Timed Petri Net (TPN) presentation that can be constructed by MOVAT from the UML Message Sequence Chart as shown in Figure 1. Failure modes detected by the TPN are then incorporated into the FMEA.

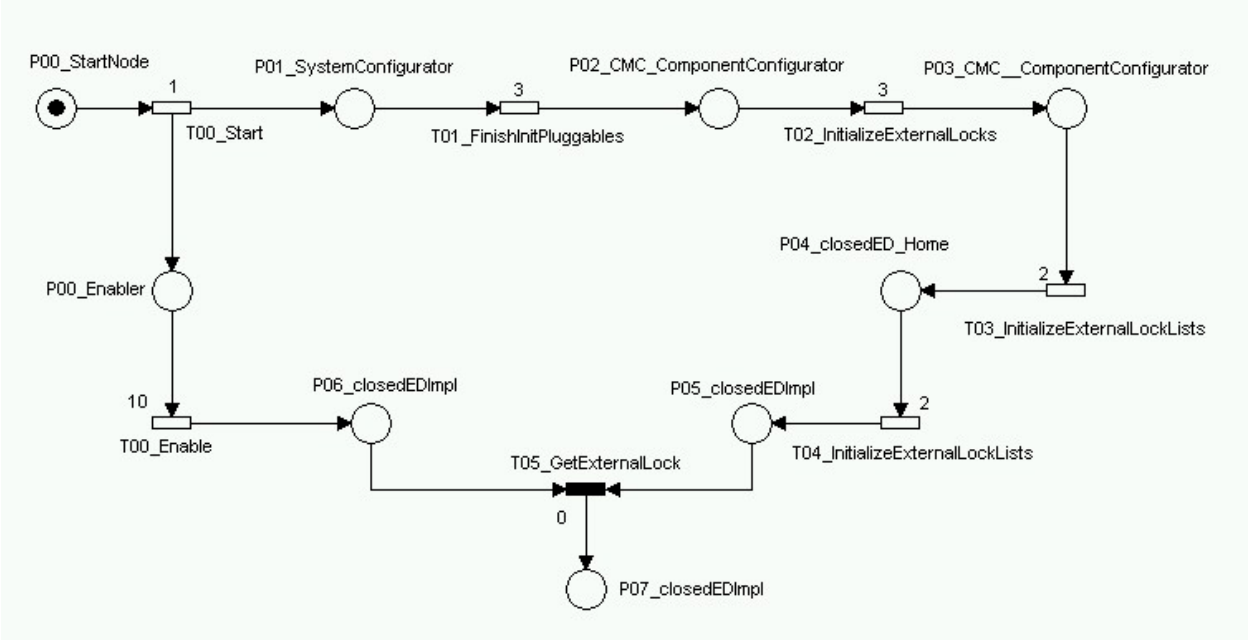

Figure 1. Example of Timed Petri Net

Black rectangles represent immediate transitions, white rectangles timed transitions, the numerals denoting the delay in units of computers cycles. Where two arrows go into a transition both inputs must present tokens in order for the transition to take place. Circles are "places", methods that execute when a token (black dot) reaches them. In this example the multiple paths entering T05 indicate an opportunity for timing conflicts that can be investigated analytically or by simulation.

3. Benefits

The FMEA format provides a systematic way of presenting potential failure modes of a program and defenses against these (detection and recovery). When all *methods* 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 Case Charts and Message Sequence Charts reduces both labor and the potential for errors and omissions.

The principles of this approach can be applied to non-UML programs by structuring these in Use Case format but the benefits of automation are then lost. The Use Case format is also valuable for preliminary V&V activities in UML based development before the program structure is complete.

B. DETAILS FOR PRE-CODING PHASE

1. Objectives of V&V prior to Coding.

The emphasis for V&V activities prior to coding is on prevention of failures that could result in unsafe conditions or that could prevent completion of the mission. At that point in time any software construct 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 Preliminary Hazards Analysis (PHA) that should have been generated in the Requirements phase.

2. Early Stage Activities

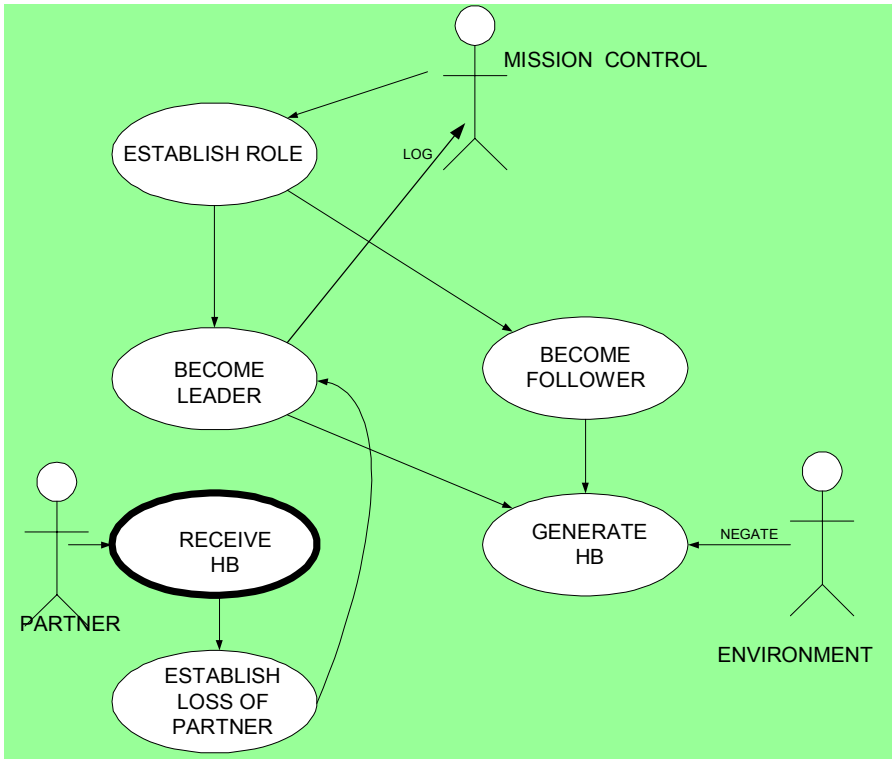

Figure 2 Use Case Diagram

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 unmanned aerial vehicles (UAVs). The ovals denote software modules (cases or methods) and the arcs show potential propagation of failure effects. 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. With some changes in nomenclature the same use case diagram can represent activation of the stand-by element when the active one fails in any system that employs stand-by redundancy. It therefore depicts a frequently encountered structure. A switch in roles is initiated when the stand-by 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 reproduced in Table 2, a simplified version of Table 1.

Table 2. Failure Modes and Effects for *Receive HB* Module.

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**

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) and due to an active failure in the *Receive HB* module this is not recognized the follower will miss becoming the leader, defeating the redundancy scheme and causing mission failure.

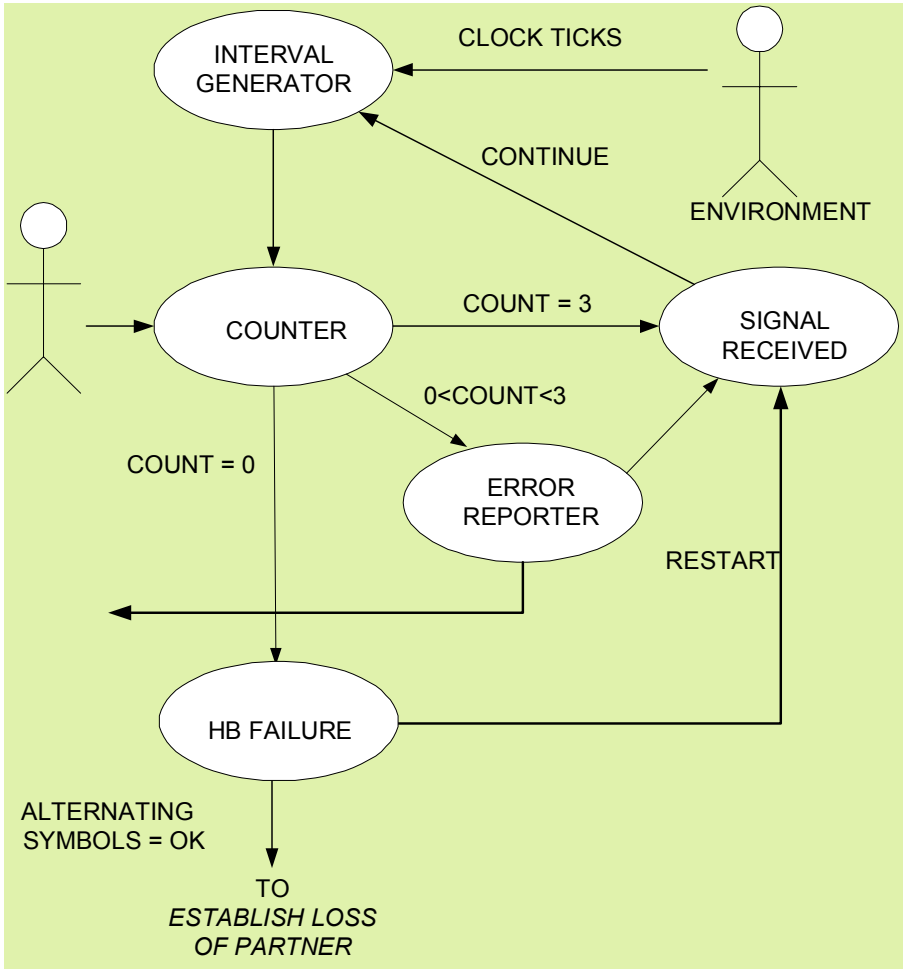

Figure 3 Detail of *Receive HB*

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.

3. Conclusions for Early Stage V&V

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.

C. FMEA Generation from Class Diagrams

1. Objectives of FMEA Generation

Class diagrams represent the structure of the software and list all of the *methods* utilized by each class. The listing of the methods represents the entry point for FMEA generation, and because all methods 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 methods. The system level severity of the effects is characterized by standardized scales, I – IV for military use (highest severity = I) and 1 – 10 for automotive applications (highest severity = 10). 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.

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 timed Petri nets, is proposed that is described in the next section of this paper.

2. Activities for FMEA Generation

The activities include the following

a. Collection of class and method data from the UML tool
b. Assignment of failure modes for each method
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 4

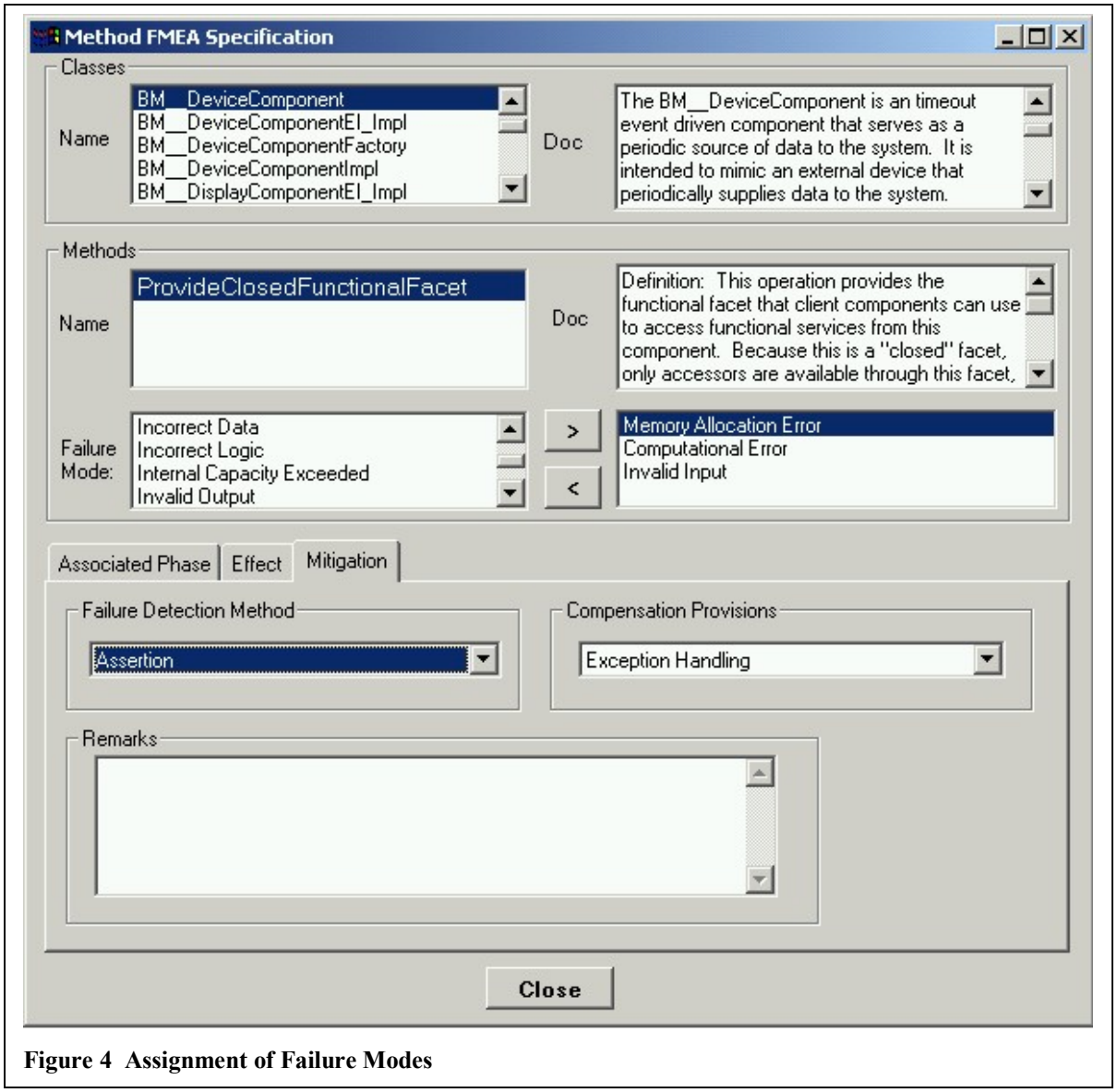

Figure 4 Assignment of Failure Modes

.

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 method 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 5. The top part of the screen is identical with Figure 4 but the lower part shows the response to the Effect and Mitigation tabs.

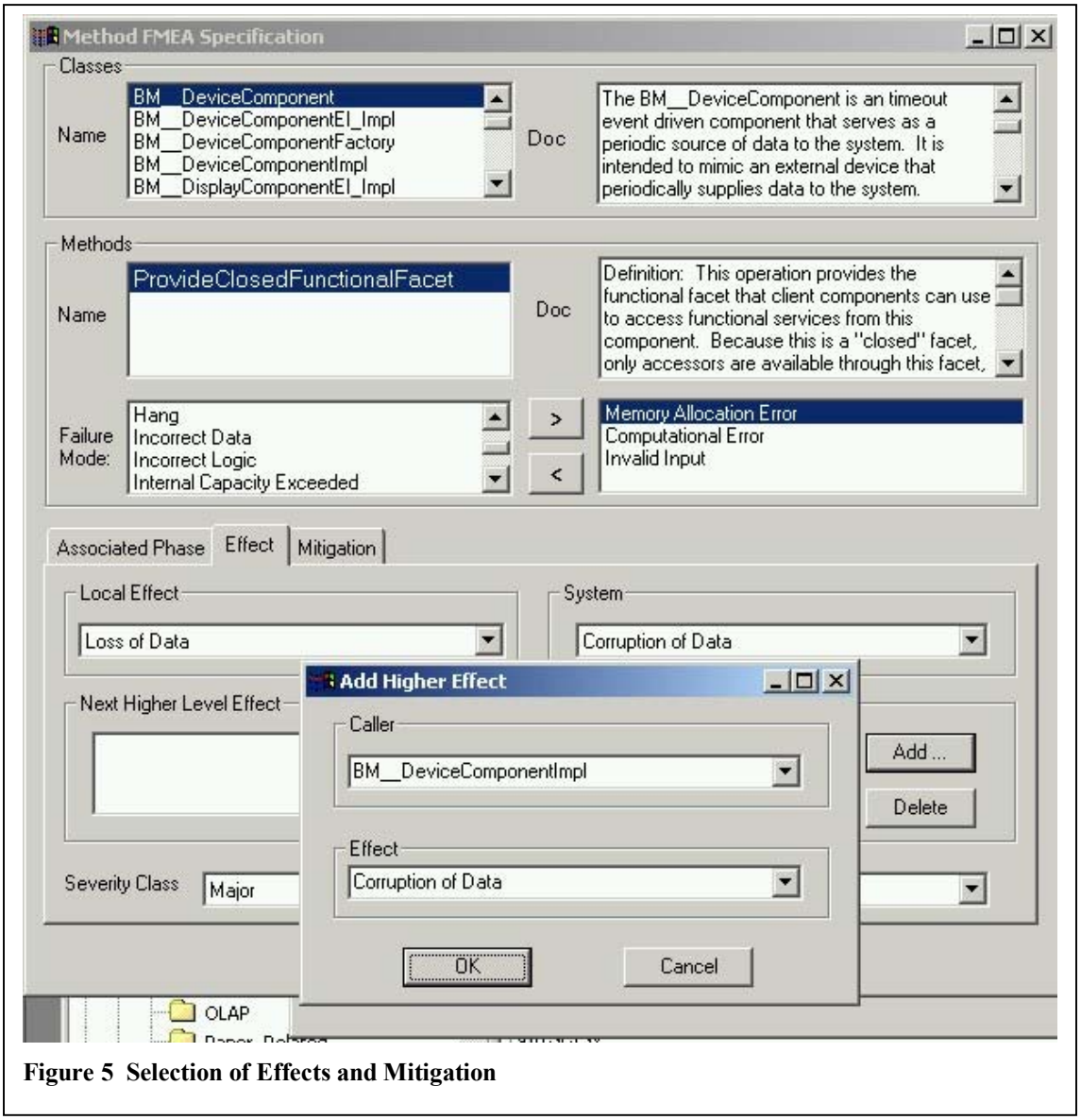

Figure 5 Selection of Effects and Mitigation

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 methods in UML and they can be tagged as detection methods. This facilitates (a) semi-automated association with the methods that are being protected, and (b) protecting the methods 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 methods that implement these features can be tagged and thus the recognition of compensation for the covered failure effects can be automated.

3. Conclusions for FMEA Generation

A major benefit is that the FMEA is based on a file of methods 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 methods 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”.

D. Timed Petri Net (TPN) Generation

1. Objectives of TPN Generation

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.

2. Activities for TPN Generation

The essential activities are:

a. Selection of collaboration charts from the UML files
b. Transcription of the UML file into MARIA format
c. Generation of a raw TPN from the MARIA file
d. Editing of the TPN to permit analysis of timing problems

An example of a collaboration chart is shown in Figure 6, which represents the exchanges required to generate a file 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.

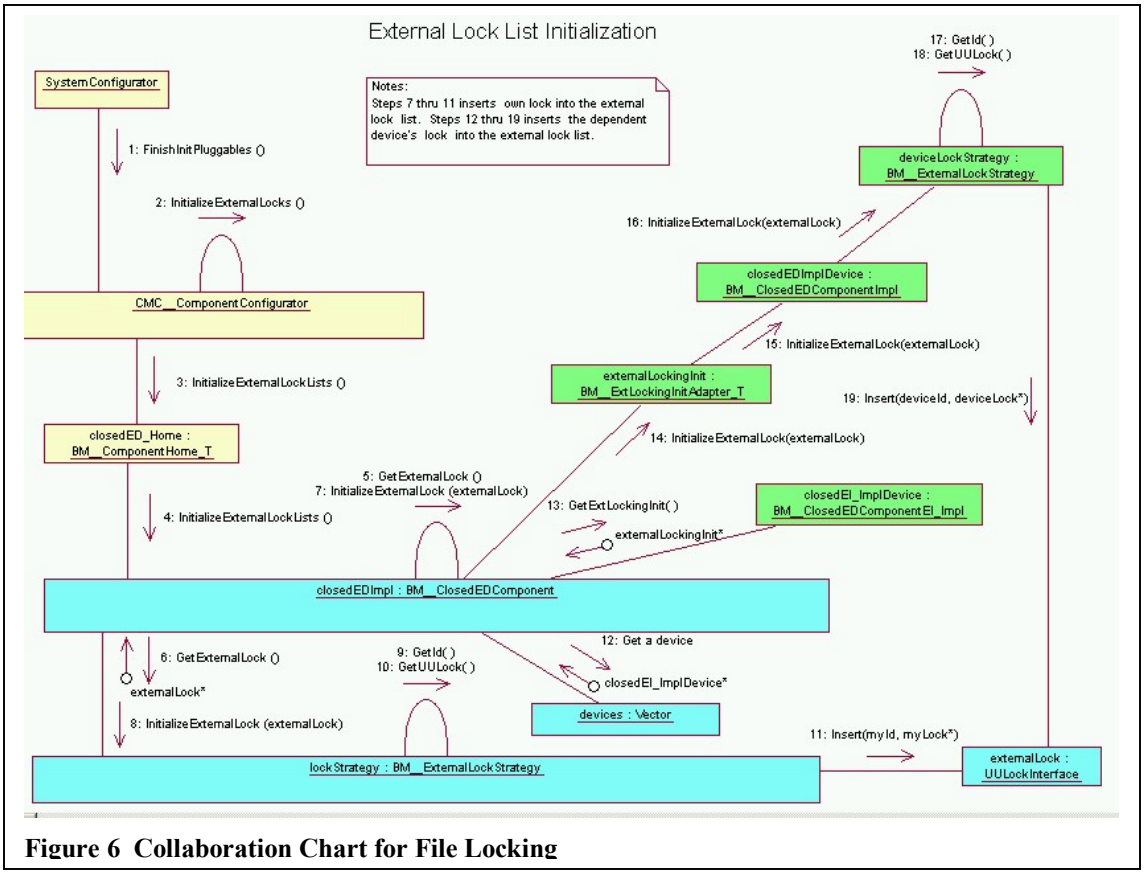

Figure 6 Collaboration Chart for File Locking

The information for generating this graphic is maintained in the UML tool as a text file that denotes the sequence in which classes and methods interchange messages. This text file can be transformed into the source file for a TPN with the aid of the MARIA (Modular Reachability Analyzer) program, a public domain product originated by AT&T that is now administered within the GNY consortium. As part of that transformation UML classes and methods become TPN places, and UML arrows become TPN transitions. A segment of the MARIA file generated from the above collaboration chart is shown in Figure 7.

```
/* Maria Model Generated by
Model Based Verification and Assessment Tool (MOVAT) */


/* Collaboration Name: BM__Component */

/* Interaction Name: External Locking Initialization */
/* Message Name: FinishInitPluggables ()
Sender: SystemConfigurator          Receiver: CMC__ComponentConfigurator*/
place SystemConfigurator bool: 1;
place CMC__ComponentConfigurator bool: 1;
place FinishInitPluggables___ bool: 1;
trans FinishInitPluggables____Success
            in  {
           place SystemConfigurator: 1;
           place FinishInitPluggables___:1; start: 1
           }
            out {
           place CMC__ComponentConfigurator: 1;
           }
;
```

Figure 7 Excerpt of Maria File

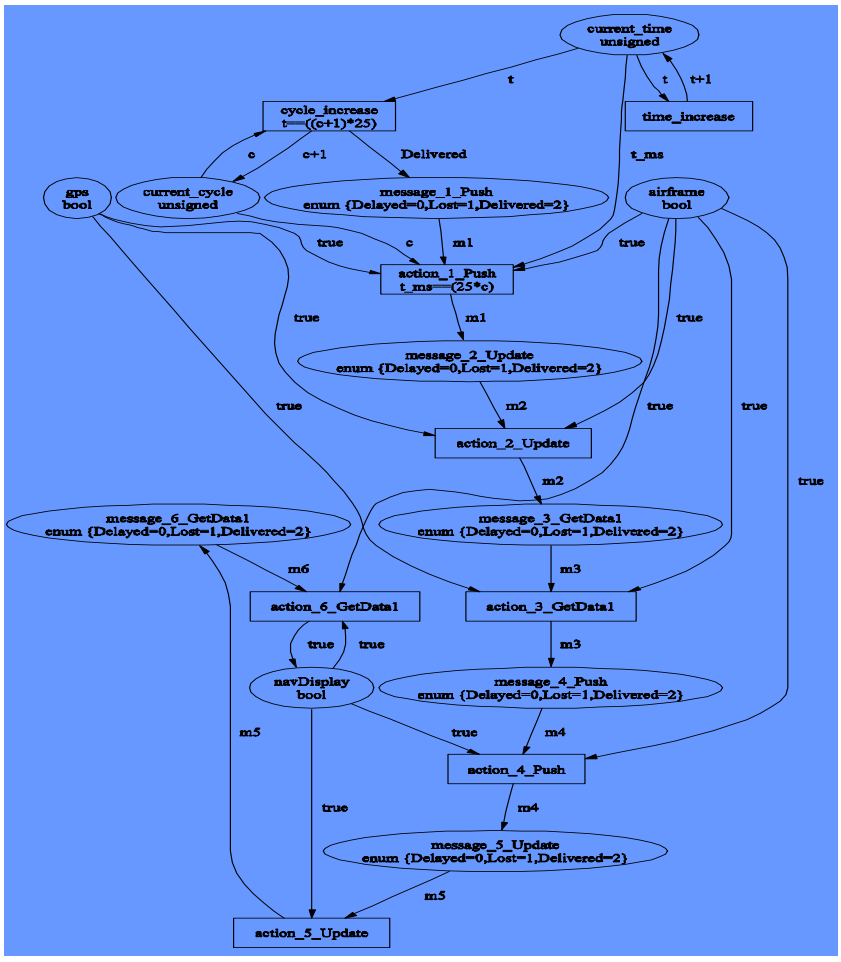

Figure 8 TPN corresponding to Figure 6

The MARIA file is in turn used to create a graphical version of the TPN as shown in Figure 8. In this representation ovals designate places and rectangles denote transitions. Arrows show information flow but not necessarily time. Figure 8 is a direct equivalent of the collaboration chart shown in Figure 6. We call this a raw TPN because in this format it is difficult to discern transitions that may lead to timing problems. The edited TPN of Figure 9 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.

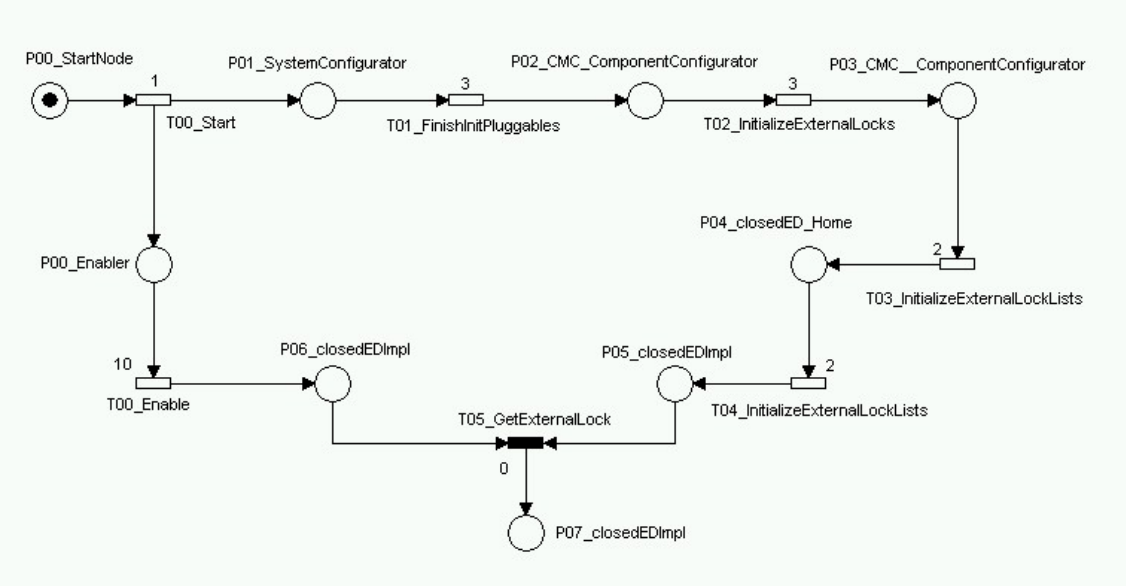

Figure 9 Edited TPN

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.

3. Conclusions on Timed Petri Nets

Because timing problems are not obvious in the MOVAT FMEA generation described in Section C, 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 Chart that is part of the UML file. Thus MOVAT includes a very powerful tool for detecting and correcting timing problems

E. Putting It All Together

The emphasis in the foregoing has been on the mechanics of generating an FMEA or a TPN. Now we address how these steps relate to better Verification and Validation of computer programs for critical applications. Better here means

- More complete
- More objective
- Requiring fewer resources
- Being an organic part of the software and therefore updated as it changes

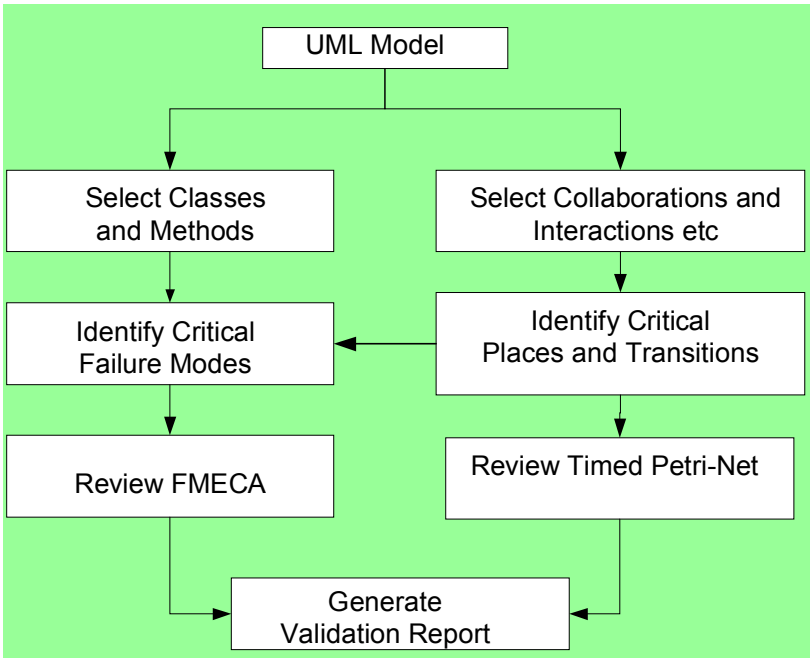

Figure 10 Using MOVAT

The process that accomplishes this is shown in Figure 10. The left branch of the figure corresponds the generation of the FMEA as described in Section C above, and the right branch corresponds to the generation of the Petri nets for identifying timing problems. As these are found, they are entered into the FMEA as shown by the left pointing arrow in the center of the figure.

The reviews (lower pair of boxes) are facilitated by the severity rankings and remarks on the FMEA as shown in Table 1 at the beginning of this paper. Instead of having to analyze the correctness of each element of the program, the reviewer can concentrate n the effectiveness of the protective measures, detection and compensation, that are entered in the table. 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 method used in the program assures completion of the process.

ACKNOWLEDGEMENT

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.