

PRODAF

A Probabilistic Software Tool Subjecting Component Designs to System-Level Reliability Constraints

Volume 1: Executive Overview

**William Strack
Sankaran Mahadevan
Satchi Venkataraman
Vinod Nagpal**

**N&R Engineering
6659 Pearl Road, Suite 400
Parma Heights, Ohio 44130**

Submitted to

**NASA Glenn Research Center
21000 Brookpark Road
Cleveland, OH 44135**

March 31, 2008

Summary

This document contains a general description of the *PRODAF* (Probabilistic Design and Analysis Framework) software tool. *PRODAF* implements a practical, multidisciplinary, design-for-reliability methodology for aerospace systems. It allows system-level reliability constraints to impact component-level designs and interfaces user-selected, physics-based deterministic modeling codes with a Fast Probability Integration code to obtain high-fidelity probabilistic component failure rate data. The computed component failure rates are input into a system-level probabilistic risk assessment code such as *QRAS* or *SAPHIRE*. A feedback loop from the risk assessment tool to the analysis/design tools enables the system-level reliability constraint to affect the component design. Accuracy measures of the probability calculations (confidence intervals) are provided to account for uncertainties in the uncertainty parameters. Design variable optimization is accelerated through the use of adaptive response surface modeling. On-screen results are presented graphically in terms of CDF/PDF plots and sensitivity charts. Two companion volumes document the mathematical methods utilized within the code as well as a detailed set of instructions to install and run *PRODAF* (Refs. 1 and 2).

Introduction

NASA's vision of future space exploration entails very challenging long duration missions that are certain to entail the pursuit of advanced concepts and technologies. Because such long duration missions will necessarily utilize design concepts and technologies that are relatively immature, they will involve substantial risks that cannot be fully mitigated through normal ground-based and short-duration flight test programs. Predictive tools will play a more important role in evaluating a design's characteristics such as performance and reliability. Consequently, it is important that designers model and analyze each hardware component using the best physics-based simulation codes available. However, even the best codes have limits to their accuracy. In addition, it is very difficult to interpret results from many different codes and evaluate the impact from a system perspective. What is needed is a method for capturing the effect of uncertainties at the component level and rolling them up to the system level to generate probabilistic risk/reliability assessments (PRA).

Recent space flight failures have heightened awareness of the need to strengthen our analytical tools to cope with these challenges. There are 2 major reasons that traditional design methodologies are inadequate: (1) existing deterministic methodologies and associated codes fail to properly account for the inevitable uncertainties inherent in unconventional/novel systems, and (2) a practical mechanism to analytically generate physics-based probabilistic component-level failure rates and collectively roll them up to the system level does not exist. Consequently, designers often resort to the old factor-of-safety approach that is generally unsatisfactory since it yields either an overly conservative design (resulting in excessive weight and cost) and/or a design whose system-level failure rate is simply unknown. As structural designs and the materials they utilize become more complex (e.g., composite materials, multifunction structures) these shortcomings will become increasing severe.

Engineers addressing this challenge today must use a variety of software tools to accomplish their tasks. These tools are powerful, expensive, and frequently provide more capability than the engineer really needs. Moreover, unfamiliar tools too often require several weeks of training, and perhaps several months of actual use to become proficient. The casual user of such tools is usually discouraged because the learning effort is too great and he spends more time learning how to use the tool than in applying it to a problem. These barriers are compounded dramatically when the task requires multiple codes and the transfer of data between them such as during many design processes.

The shortcomings above have been addressed over the years by various research efforts (see, for example, Refs. 3-5). In most cases, methods and software are demonstrated that allow existing discipline-oriented design/analysis codes to be used in an integrated fashion. The codes are re-engineered to have precise interfaces defined for what they need (inputs) and what they produce (outputs). Software is provided to link the codes together and automate data exchange procedures. Frequently, a collection of modules is also provided to support optimization. These include numerical, heuristic, and other non-numerical optimization methods.

The combined capabilities described above are often referred to as an “environment” or “framework”. A detailed discussion of the ideal characteristics of a multidisciplinary framework to support design, analysis, and/or optimization is beyond the scope of this paper. However, there are excellent studies of these requirements in the literature (Ref. 6). There are also numerous demonstrations of prototype environments. However, most do not support probabilistic risk assessments.

The *PRODAF* suite of codes is the result of recent efforts to provide a practical, multidisciplinary, probabilistic design and analysis framework. The framework supports integration of existing codes – thus eliminating the large learning investment of a start-from-scratch approach. That is, it provides a graceful transition from current practices to a multidisciplinary, physics-based probabilistic methodology while maintaining the confidence of designers.

PRODAF is intended to accelerate the design/analysis process, take the grunt work out of the typical engineering tasks of transferring/converting data streams from one application code to another, and capture the intrinsically probabilistic nature of design problems. This capability enables engineers and managers to spend more of their time interpreting results and making wise decisions as well as yield a physics-based design-to-reliability solution.

The scope of design problems addressed by the *PRODAF* tool suite ranges from the detailed design/analysis of a single part to the conceptual design of an entire system that consists of many interacting subsystems, components, and parts (mechanical, fluidic, electrical, et cetera). The ability to utilize physics-based, high-fidelity, comprehensive models to obtain accurate loads and environmental conditions is a major step forward. This capability allows trouble spots to be detected much earlier in the design process and can significantly reduce subsequent redesign and experimental testing costs.

PRODAF provides an integrated design environment enabling aerospace hardware designers to rapidly determine optimum risk-constrained designs subject to multiple uncertainties in applied loads (aero, thermal, mechanical), material properties, and manufacturing processes. This means that the design process no longer consists of a sequence of separate code invocations to: (1) obtain the geometry model, (2) determine the various loads, (3) determine performance, (4) perform a structural analysis, (5) perform design optimization, and (6) perform a probabilistic risk assessment. Instead, all of these functions would be automatically incorporated into a single framework using existing physics-based deterministic modeling codes and a set of computer-generated data transfer interfaces. Thus, a design engineer would be able to rapidly explore the design space to identify the minimum weight design that meets a given reliability constraint – thereby avoiding both an overly conservative design and an excessively risky design. For example, the wall thickness could be determined of a launch vehicle’s external cryogenic propellant tanks exposed to uncertain thermal and aerodynamic loads as well as uncertain material properties and subject to a specified mission reliability.

PRODAF Description

PRODAF makes no assumptions about design processes. It does not use any particular design process; rather, it assumes that different users will want to define their own design process. The conceptual design process for a new spacecraft, for example, differs radically from the design process for a production component or system. Hence, *PRODAF* provides a simple graphical tool that enables the user to architect his own design process simulation. In general, this means that users will select which modeling and analysis codes are to be utilized in the overall architecture and the data transfer linkages between them.

An example such design architecture is illustrated in Figure 1. To begin, a set of mission requirements and initial design variable values are defined (the yellow ellipses). These are utilized by a CAD code to produce an initial geometry model. Note: software codes are represented by the blue boxes and data output files by the gray boxes. The initial geometry model serves as input to various component analysis codes that may be either physics-based or empirical-based. These deterministic codes are linked to a generic probabilistic code (either the FPI Fast Probabilistic Integration code or a Monte-Carlo code, denoted by the orange boxes) to generate the model-based reliability data needed by the *QRAS* (or *SAPHIRE*) system-level reliability code (Ref. 7). Manufacturing and/or life predictions codes may also be involved in order to generate probabilistic failure rates. A feedback loop interfaces the design variable set to system-level reliability (or other criteria) and thus provides a mechanism to intelligently set the design variables. An efficient optimization mechanism may also be required if the design variables are to be optimized.

This particular design process architecture is a baseline template that can be modified by users to suit their specific needs. What is important is that the methodology permits users to define the design process in terms of the sequence of codes to be used, linkages between the codes, uncertainties, and roll multiple component-level physics-based reliability results upward to a system-level Probabilistic Risk Assessment (PRA) tool. The only codes that *PRODAF* provides in this architecture are the probabilistic *FPI* code and the optimization code. All the others are user-supplied.

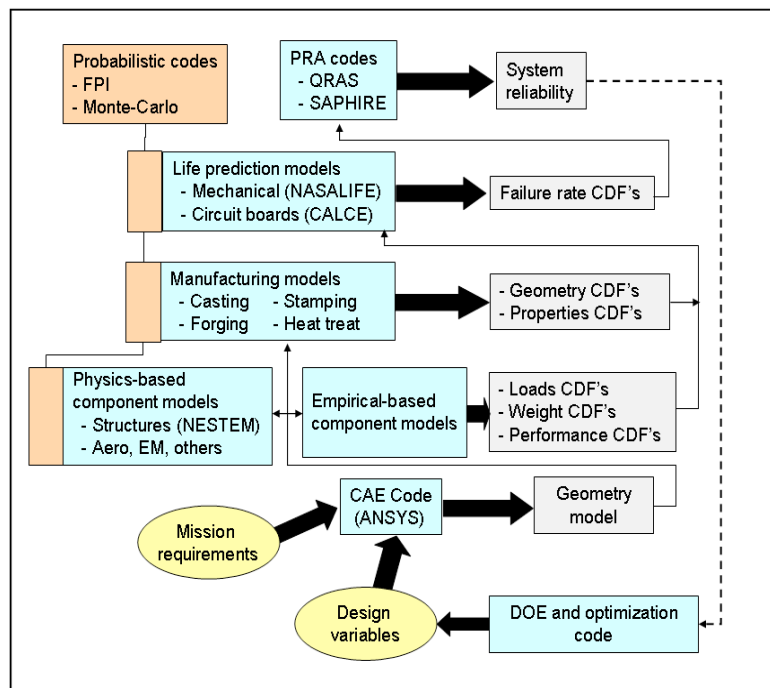


Figure 1 – Example design process architecture

The prototype *PRODAF* software suite that was developed to implement this methodology consists of a set of integrated tools – all of which are invoked from a single executive that is illustrated in Figure 2.

Code Library Manager – This tool is used to construct a decomposition of all design/analysis codes available to the user. The codes are displayed in a hierarchical tree as illustrated in the left-hand pane of Figure 2. The organization of this display may be edited into whatever categories the user defines (e.g.,

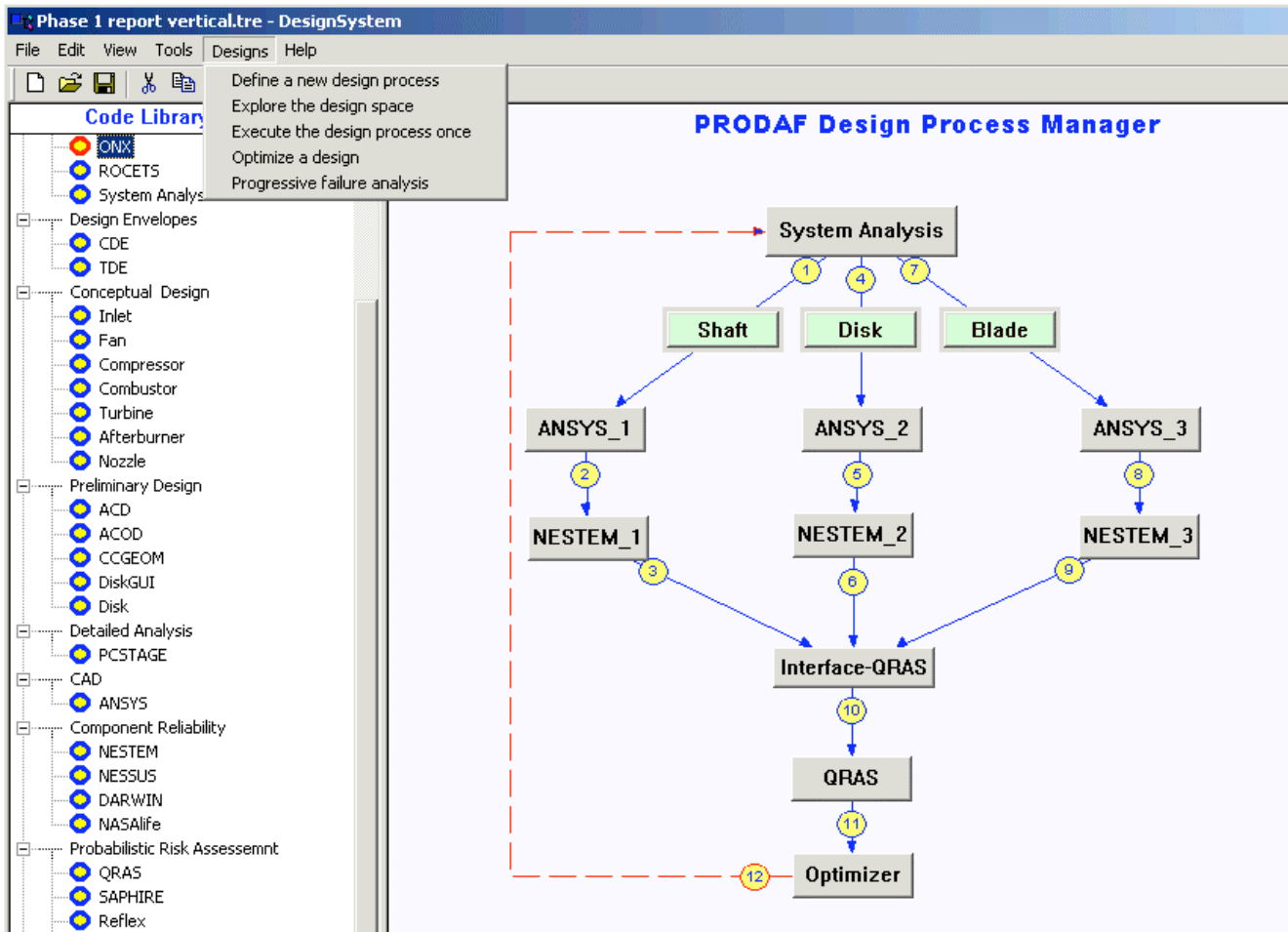


Figure 2 – The main PRODAF user interface.

Conceptual Design, Preliminary Design, Detailed Design, CAD, and System Risk Assessment). Users may add, edit, and delete entries in this tree using the context menu associated with each item, or relocate items using conventional drag-and-drop techniques. The hierarchy of codes displayed within the Code Library Manager may also be saved and retrieved. Although the main purpose of the Code Library Manager is to help users visualize the code choices they have in a single display, the codes may also be invoked using the tree itself – either through the context menu or by double clicking an item. The context menu method offers a choice of running the selected code deterministically or probabilistically while the double-click method is limited to deterministic runs. Invoking codes from the Code Library Manager is facilitated by a database that links the code names to user-selected executable files as well as input/output file names and any command-line options. This means that the codes in the library need not be located in one central directory; rather, they can be located wherever users happen to have them (even on remote servers). Additional information is added to this database to facilitate probabilistic invocations; specifically, the name of an uncertainty configuration file that defines the uncertainty parameters, if any.

Design Process Manager – This tool enables users to configure the design process; that is, select which specific codes will be used and the linkages between them as well as visualize the configuration on-screen as illustrated in the right-hand pane of Figure 2. Codes selected by the user from the left-hand library pane may be added to the design process diagram in the right-hand pane using the context menu. The code symbolic representations (the gray boxes) may be repositioned and resized simply by dragging the icons

using the left mouse button. Just as in the Code Library Manager, any of these codes may be invoked either deterministically or probabilistically using a context menu. They also may be linked (or unlinked) to any of the other codes using the same context menu. These linkages indicate data transfer paths within the design process and imply a behind-the-scenes interface to accomplish the transfer. Labels may be added to indicate the associated components. Similarly, numbers may be added to the links (displayed within circles) to specify the execution sequence.

The example displayed in Figure 2 represents a relatively simple design process for a 3-component rotor assembly consisting of a shaft, a disk, and a set of 24 blades. The geometry for each component is generated by *ANSYS*, the structural modeling and analysis is performed by *NESTEM*, and the system reliability is determined by *QRAS*. An interface module is also required to handle the data transfer from *NESTEM* to *QRAS* since *QRAS 1.7* does not support user-supplied input and output files (although it does save and restore user data supplied via its GUI in binary database tables). This interface module enables users to modify the *QRAS* binary database tables.

Probabilistic Manager – If a user wishes to conduct a probabilistic analysis at some stage of the design process, he may invoke the Probabilistic Manager tool. This tool enables users to define and characterize uncertainties associated with variables within text input files for a specific code. It automatically creates and runs a script file that invokes the code multiple times to obtain the probabilistic response data required in the probabilistic analysis performed by the *FPI* code. Any executable code or Excel spreadsheet is a candidate for the Probabilistic Manager. Once the probabilistic data has been generated, *FPI* may be invoked to generate and display the results in the form of CDF/PDF curves and probabilistic sensitivity charts.

Figure 3 summarizes the overall probabilistic analysis process implemented in *PRODAF*. In essence, a set of perturbation runs of a user-selected simulation code are harvested for their response variables in order to obtain the input required for *FPI*. *FPI* uses this data to perform a probabilistic analysis. Figure 4 displays a typical Probabilistic Manager user interface for handling uncertainties associated with a typical deterministic simulation code (the *Launch_Vehicle* code estimates the payload mass and fuel mass for a fixed gross takeoff weight launch vehicle). The uncertainties are enumerated in the table in the upper-left together with their mean values, coefficients of variation, and types of distribution. Response variables are listed in the lower left table (two in this example).

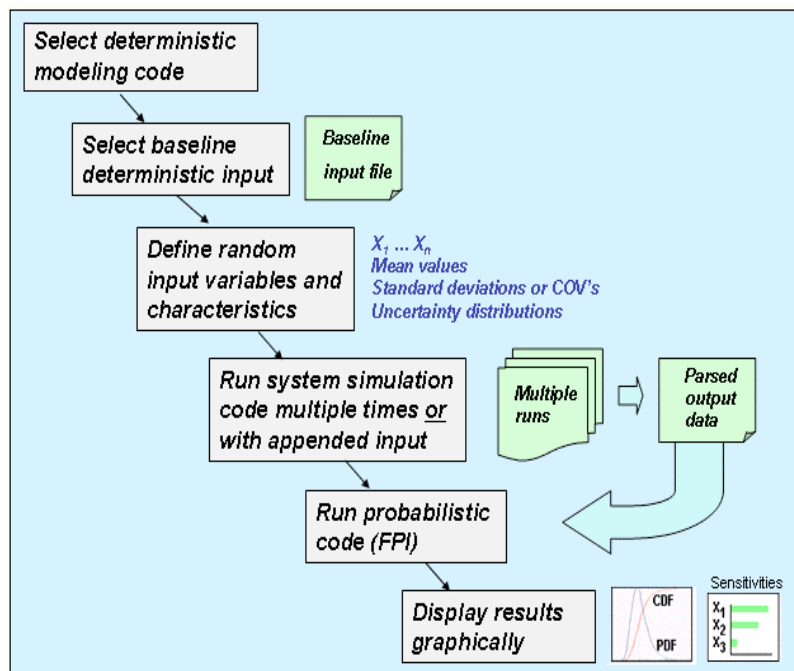


Figure 3- *PRODAF*'s probabilistic analysis process.

Users are offered several methods to specify the uncertainty characteristics and inform *PRODAF* how to locate these variables in the input file(s) and locate the response variables in the output file(s) as illustrated in the right-hand portion of Figure 4 for one uncertainty (tank mass ratio). Many legacy deterministic codes use labeled input and output files making it convenient to specify uncertainties using

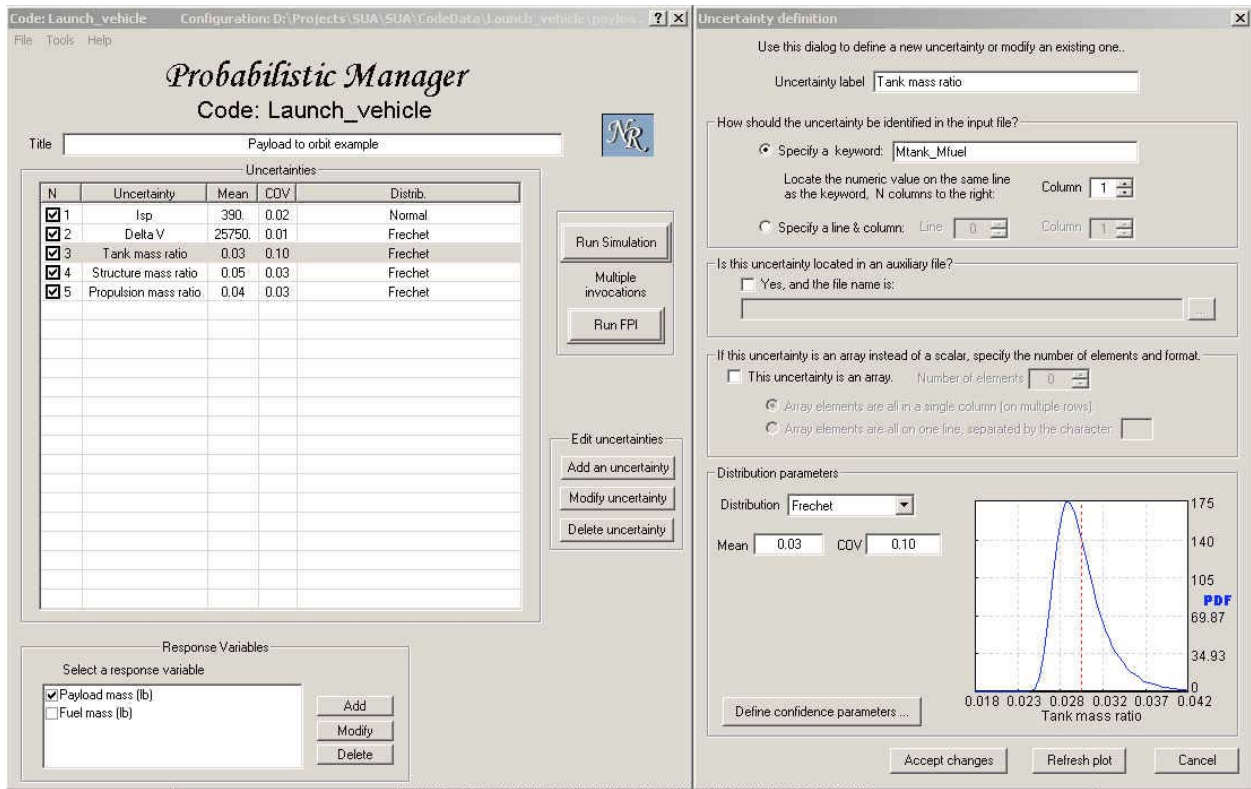


Figure 4- Probabilistic Manager tool illustrated with a typical example application code.

keywords. However, some codes use fixed-field formatted input/output files with no labeling – just lines of numbers. Such codes require specifying the uncertainties and response variables in a row-column format.. Array variables representing curves are handled in addition to scalar variables, and auxiliary input files may also be specified. The lower portion of the uncertainty definition dialog (see Figure 4) is used to set the distribution type and its parameters, and to graphically display the PDF curve for each uncertainty. Current distributions include: Normal, lognormal, Weibull, Frechet, Beta, modified Beta, truncated Normal, truncated Weibull, uniform, maximum entropy, extreme value, and user-defined polynomial curve. Distribution-specific parameters are specified by the user in the edit boxes and these are transformed by *PRODAF* into equivalent mean and COV values that are displayed in the main uncertainty window. PDF plots may be displayed so users can visualize the impact of contemplated parameter changes.

A similar dialog is used to define response function variables. Once all the uncertainties and response functions are defined, users may ask *PRODAF* to generate a dataset of responses that will subsequently be used by the *FPI* code to calculate probabilistic results. The response dataset is produced by invoking the simulation code

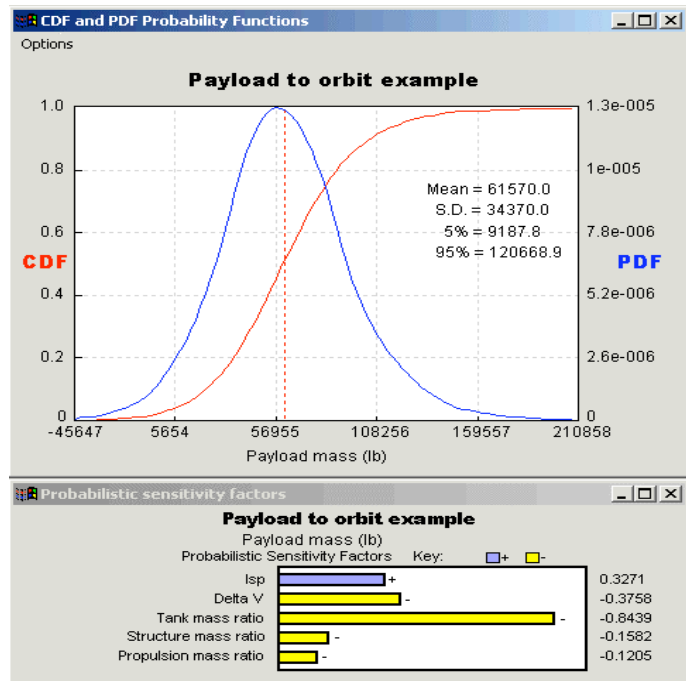


Figure 5 – Probabilistic analysis results charts.

(*Launch_Vehicle* in this example) multiple times – a process that is controlled automatically by *PRODAF*. Subsequently, the *FPI* code may be invoked to perform a probabilistic analysis and display the results as shown in Figure 5.

Both CDF and PDF curves may be plotted as well as a probabilistic sensitivity chart that indicates which uncertainties have the greatest impact on the response function(s). Users are offered considerable flexibility in defining the methodology used in the probabilistic analysis:

- Linear or quadratic limit state function regression approximation
- Positive, negative, or central difference perturbations
- Coefficient-of-variation (COV) or standard deviation specification of the uncertainties
- Display of traditional deterministic sensitivities or probabilistic sensitivity factors

The Probabilistic Manager is intended to be used at whatever stage within the design process the user feels it would be helpful to generate probabilistic results.

Confidence Intervals – In many probabilistic problems it is difficult to obtain the required statistical parameters used to characterize uncertainties. This is frequently the case when investigating new concepts or advanced technologies since little, if any, relevant empirical (or even analytical) statistical data exists. In such cases, users are forced to estimate or guess values that they believe are reasonable. However, this means that there is uncertainty about the uncertainties and it is desirable to be able to indicate the degree of confidence in the resulting CDF and PDF curves. This may be accomplished in *PRODAF* using confidence interval determination techniques that are described in the companion *PRODAF* Theoretical Manual (Ref. 1). To include confidence interval determination, users must provide additional information to characterize the uncertainty in some or all of the primary uncertainty parameters. Determining confidence intervals requires a substantial increase in computational time since many more simulation code invocations (and *FPI* invocations) are required. The result is a pair of auxiliary CDF curves that bound the baseline CDF curve as illustrated by the green band in Figure 6. The red curve is the traditional CDF curve based on the expected uncertainty parameters, and the two dashed blue curves represent the upper and lower limits (typically the 5 and 95 percentiles) of the CDF determined by the uncertainties in the uncertainty parameters. Such a plot can reveal to designers and managers how much variation might occur in the system-level merit criterion due to not knowing the uncertainties with definiteness.

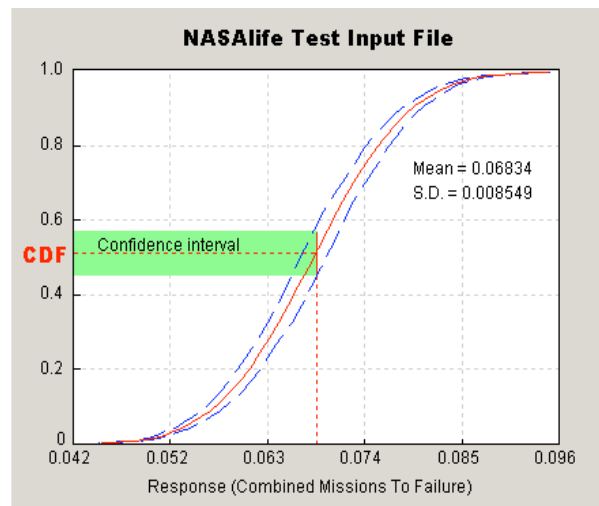


Figure 6 – Adding confidence intervals to CDF plots.

Three methods are offered to characterize the confidence interval uncertainties:

- Users may specify the distribution type and supply the concomitant parameters
- Users may provide a set of test data and *PRODAF*'s Probabilistic Manager will determine both a

suitable distribution type and its parameters

- Users may specify a distribution and *PRODAF* will determine the uncertainty parameters from a set of sample data

Design variable optimization – *PRODAF* includes a design variable optimization methodology that enables users to explore the design space efficiently. Many physics-based modeling codes involve lengthy calculations, so it is important to utilize search strategies that are faster than typical brute-force methods even if they sacrifice some accuracy. Response surface modeling (Figure 7) is an attractive technique for such situations provided there is not a large number of design variables. An efficient response surface technique to locating optimal design points was implemented in *PRODAF* that progressively shrinks the modeling region (called “trust region”) as the extremal point is approached. This global-local technique employs an optimum symmetric Latin hypercube algorithm to select the points used to construct the response surface. The details of this adaptive search methodology are contained in the *PRODAF* Theoretical Manual (Ref. 1).

The *PRODAF* code implements this algorithm in a general way; that is, the user may select the function evaluation code, identify and characterize the design variables, and select values for the optimization control parameters via an “*Optimization Manager*” dialog.

A sequential quadratic programming (SQP) method is used to locate the optimum point within each trust region. That is, the problem is to:

$$\text{minimize } f(x)$$

subject to:

$$g(x) = \sqrt{x \cdot x^T} = 0$$

$$x_{lower} \leq x \leq x_{upper}$$

where x is the design variable vector with upper and lower bounds, and $f(x)$ is a user-specified response function such as weight, cost, performance, or failure probability

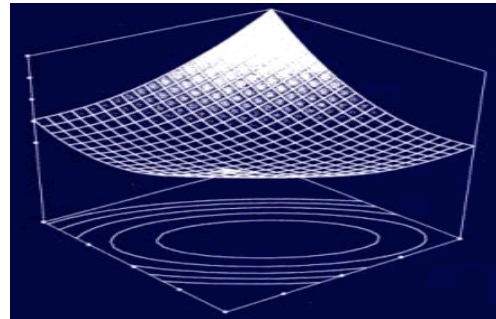


Figure 7 – Response surface modeling.

Data exchange interfaces – Although *PRODAF* establishes a framework to enable probabilistic design processes that involve a sequence of user-defined simulation codes, a dilemma still exists; namely, how to implement a mechanism to transfer data between the codes. Since *PRODAF* is a generic framework that permits a wide range of user-specified codes and each code pair requires a unique interface, *PRODAF* cannot provide a generic code-to-code data transfer interface. These interfaces are code-specific and

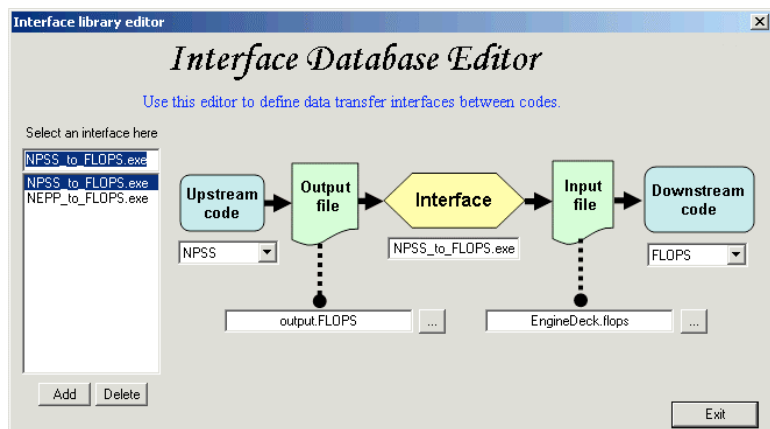


Figure 8 – The Interface Database Editor helps users setup the data transfer mechanism that connects codes.

must either be acquired externally or written by the user. However, a dialog is provided by *PRODAF* to help users setup the data transfer linkages (Figure 8).

In some cases these interfaces will be relatively easy to implement with straightforward scripts, but in other cases they will be more challenging. Simulation codes that are entirely encapsulated such that all input and output are implemented through a GUI are particularly difficult to integrate into *PRODAF*. *QRAS 1.7* provides an example of a difficult interface since it is driven through a GUI only (e.g., no provision for user-defined input files). However, since it does archive the GUI-generated data in an undocumented binary database, a “backdoor” approach can be used to create an interface. This approach was implemented for one of the mainstream *QRAS* GUI-input formats that is compatible with *FPI* and *CALCE* output. (*CALCE* is a physics-based electronic circuit board probabilistic reliability code). The *QRAS* Interface Manager (Figure 9) harvests probabilistic component reliability data embedded in *FPI*-generated or *CALCE*-generated output files, displays component failure probabilities, and incorporates the failure probability data into the system-level *QRAS* binary database files. In the example shown, three electronic components and two mechanical components are represented. The columns with headings “Phase”, “Start time”, and “End time” contain mission duty cycle data associated with the predicted failure probabilities.

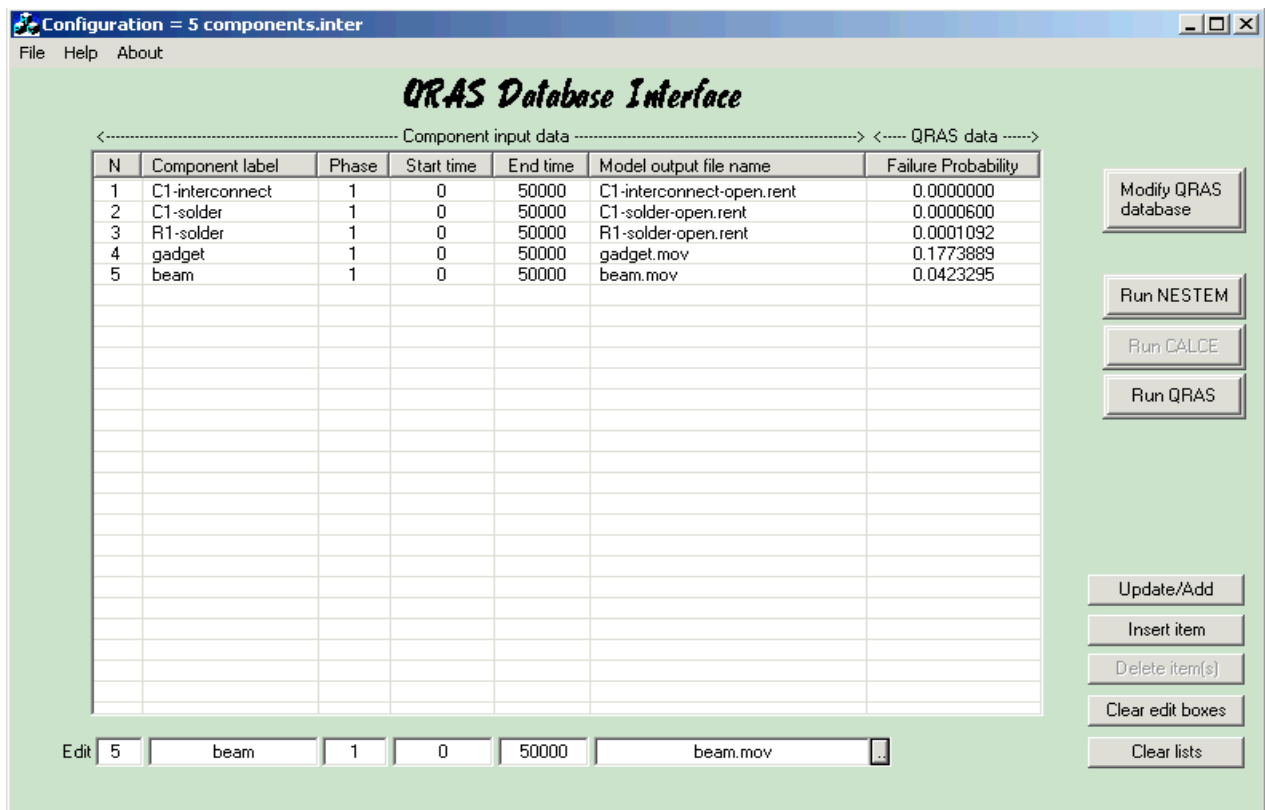


Figure 9 – *QRAS* Database Interface dialog.

Whenever the user requests a design process simulation, *PRODAF* first checks to make sure all of the interfaces are available before initiating the simulation. If any modeling, analysis, or interface codes are missing, *PRODAF* advises the user of the missing elements and halts.

Convenience tools – Several auxiliary software tools are included in *PRODAF* for the sake of convenience. One is a text editor that may be used to modify text input files and view output files. This tool is automatically invoked whenever a probabilistic analysis is requested via the Probabilistic Manager and displays the main deterministic input file. Another tool generates a random sample of points given a distribution type and associated parameters. It displays these points graphically (see Figure 10) as well as writes them to an output file.

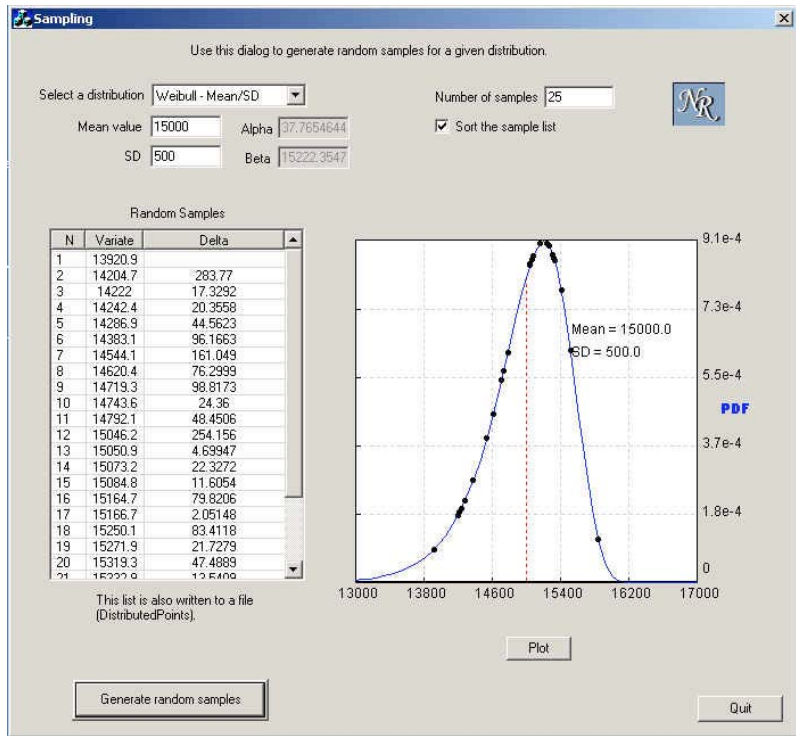
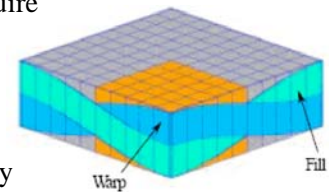


Figure 10 – Statistical sampling tool.

Physics-based progressive failure modeling of complex systems – Failure processes for complex structural systems frequently involve more than just a single component failure. Rather, a sequence of several component failures eventually lead to a catastrophic system-level failure. Analyzing such progressive failure processes is not straightforward and can easily require inordinate amounts of computational effort. In recent years, advanced methods have been developed to handle such problems (e.g., composite structures) more efficiently without sacrificing essential accuracy. *PRODAF* offers an advanced progressive failure methodology based on a fast branch and bound search procedure. System failure probability is computed using only the most likely set of failure sequences to attain computational efficiency. The methodology is documented in the companion *PRODAF* Theoretical Manual (Ref. 1).



Example Reliability Problem

Example problem using the *PRODAF* tool. To illustrate the functionality of the *PRODAF* tool, a simple generic 3-component rotor assembly was designed and analyzed. The assembly consists of a shaft, a disk, and a set of 24 fan blades as depicted in the finite-element models shown in Figure 11. To retain

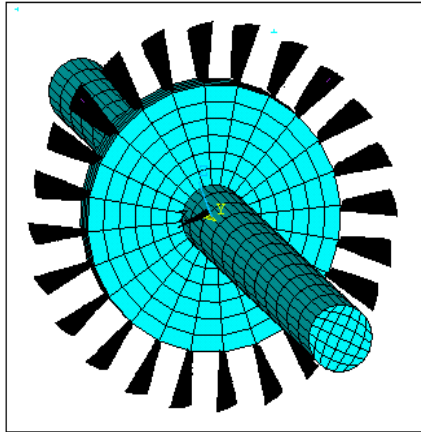


Figure 11a – Example rotor assembly FEM model.

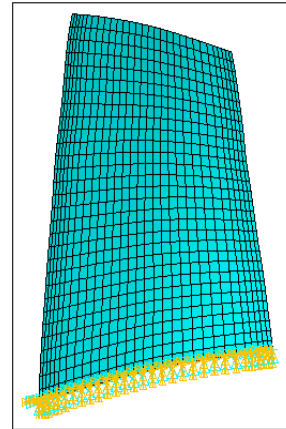


Figure 11b – Single fan blade FEM model.

simplicity yet still demonstrate the *PRODAF* design process, this example focuses on the structural portion of the total design process (i.e., the aerodynamic portion is assumed completed. Attaining a specified reliability for the entire assembly is the design objective and the stress-strength failure mode is used to determine the failure risk of each component. These components are subjected to thermal, inertia and pressure loadings. The inertia load is from rotation of the assembly at an angular velocity of 1800 rpm. The temperature profile on the blade varies from 400 °F at the root to 700 °F at the tip. The temperature of the rotor and the shaft is assumed to be constant at 600 °F.

The shaft is 72 inches long, has a 3 inch diameter, and was modeled with 864 eight-node brick elements and 1225 nodes. Both ends of the shaft are held against translation and rotation in the transverse directions. One end is held against axial translation. Inertia forces are applied at the outer surface of the midpoint in the tangential direction on the main coordinate axes.

The rotor has a 12 inch outer diameter, 3 inch inner diameter, and is 2 inches thick uniformly. The rotor model has 576 eight-node brick elements and 840 nodes.

The fan blades have a tapered chord varying from 2.2 inches at the root to 2.9 inches at the tip. Their heights linearly decrease from 2.9 inches at the leading edge to 2.2 inches at the trailing edge while their thickness varies from 0.170 inches at the root to 0.025 inches at the tip. The blade model has 900 four-node shell elements and 961 nodes.

The basic material properties were the same for all 3 components; namely:

Young's modulus, psi	Density, lb-sec ² /in ⁴	Thermal expansion coefficient, in/in/°F
29.246 x 10 ⁶	74.790 x 10 ⁻⁵	6.8 x 10 ⁻⁶

Uncertainties were assumed to be in the shaft rotation speed, the 3 basic material properties, and the strength as characterized by 9-terms in an equation that accounted for degradation due to temperature and number of load cycles. Hence there were a total of 12 random variables each of which was assigned a coefficient of variation of 0.01.

The probabilistic FEM structural analysis code *NESTEM* uses lengthy text-based input files to set up calculations of the failure probability for each component. The *NESTEM* output files were then harvested for these failure probability values. Finally, the failure data were incorporated into database tables for the *QRAS Interface* code. Since there are 24 fan blades in this example, the *NESTEM* blade failure probability was multiplied by 24 to yield a failure probability for the entire set of blades. Finally, *QRAS* was invoked to determine the system level failure probability. The *QRAS* fault tree simulation consisted of 3 phases: *Start*, *Run*, and *Stop* with durations of 1000, 49000, and 300 seconds, respectively. To retain simplicity, the failure rates for the *Run* and *Stop* phases were just assigned, while *NESTEM* provided the physics-based failure probabilities for the *Start* phase. Event sequence diagrams were defined for each component with a loss of mission defined if any component fails. The results are summarized in Table 1 and Figure 12.

Table 1 – Rotor Assembly Failure Probabilities

Component	Mean failure probability
Shaft	0.000000001
Disk	0.035110000
Single fan blade	0.001430358

System	Mean failure probability
Assembly (shaft, disk, 24 blades)	0.06818

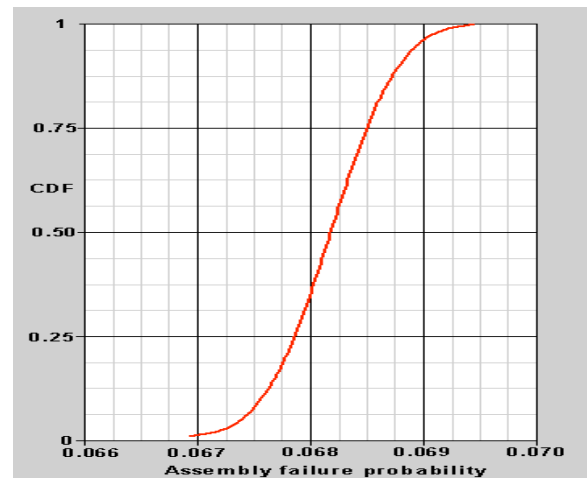


Figure 12 – CDF for the fan rotor assembly.

Concluding Remarks

PRODAF's basic functionality has been established via NASA SBIR Phase I and Phase II awards. The tool enables legacy physics-based deterministic computer codes to be coupled to a probabilistic engine in order to capture the impact of uncertainties in applied loads, material properties, geometry, and manufacturing processes on component reliability. Further, a framework was developed that permits the physics-based component reliabilities to be automatically passed upward to the system-level *QRAS* probabilistic reliability assessment code. This is the first step in a multi-step process to develop a capability to include system-level reliability as a design variable/constraint during the design process at the component level. Ultimately, the goal is to be able to rapidly explore the design space to identify minimum weight designs (or other criteria) that meet given system-level reliability constraints – thereby avoiding both overly conservative designs and excessively risky designs.

PRODAF provides an approach to identify cost-optimized designs by adjusting tolerances of the most and least sensitive variables. Tightening tolerances on the most sensitive variables increases the component cost, but may improve the reliability/cost ratio of the design. Cost may be reduced by relaxing the tolerances on the least sensitive variables.

Future enhancements to *PRODAF* can be envisioned, such as:

- Ability to separate uncertainties into aleatory and epistemic categories in order to strengthen its usefulness to programmatic decision-makers since the former cannot be reduced by further study or tests while the latter can.
- More efficient method(s) to estimate confidence bounds. The currently implemented method uses a 2-loop FPI procedure that requires 30-100 (or more) times as many implicit function evaluations as problems without confidence bounds estimation. Several approaches can be pursued to reduce this huge penalty substantially without significant accuracy sacrifice.
- Development of a set of example applications to assist users in setting up various types of problems.
- Parametric geometry and CAD interfaces to facilitate geometry input and modification.
- Grid/mesh generation and refinement methodologies that require as little user input as possible so that they can be run in an optimization loop.
- Adding a practical cost module that permits user-specified cost-estimating relationships or the use of a generic cost database that is periodically updated.
- A facility to archive previously executed problem solutions into a database that may be intelligently queried for recommended solutions (or initial design points) to similar future problems.

Finally, N&R Engineering wishes to acknowledge the important contribution of Dr. Shantaram S. Pai of the NASA Glenn Research Center to provide valuable overall guidance and support during the development of *PRODAF*.

References

1. Mahadevan, S. and Venkataraman, S., “*PRODAF*, A Probabilistic Software Tool Subjecting Component Designs to System-Level Reliability Constraints, Volume 2: Theoretical Manual”, March, 2008
2. Strack, W. C., “*PRODAF*, A Probabilistic Software Tool Subjecting Component Designs to System-Level Reliability Constraints, Volume 3: Users Manual”, March, 2008.
3. Buckley, M. J, Fertig, K.W. and Smith, D.E., “Design Sheet: An Environment for Facilitating Flexible Trade Studies During Conceptual Design,” AIAA 92-1191, Feb. 3-6, 1992.
4. Kodiyalam, S., Swenson, L.W. and Stehlin, B., “Multidisciplinary Design Optimization with Object-Oriented Product Modeling,” Engineous Software Inc., Optimization in Industry Conference Organized by the Engineering Foundation and AIAA, March 1997.
5. Krishnan, R., “Evaluation of Frameworks for HSCT Design Optimization , NASA CR-1998-208731, 1998.
6. Salas, A.O. and Townsend, J.C., “Framework Requirements for MDO Application and Development,” AIAA-98-4740, 1998.
7. Mosleh, A.; Rutledge, PJ; Groen, FJ, “Quantitative Risk Assessment System (QRAS) for Space Mission PRA”, Journal: Joint ESA-NASA Space-Flight, 2002.