

# **Architectural and Algorithmic Requirements for a Next- Generation System Analysis Code**

V. A. Mousseau  
R. W. Youngblood

May 2010



The INL is a U.S. Department of Energy National Laboratory  
operated by Battelle Energy Alliance

# **Architectural and Algorithmic Requirements for a Next-Generation System Analysis Code**

**V. A. Mousseau  
R. W. Youngblood**

**May 2010**

**Idaho National Laboratory  
Idaho Falls, Idaho 83415**

**<http://www.inl.gov>**

**Prepared for the  
U.S. Department of Energy  
Office of Nuclear Energy  
Under DOE Idaho Operations Office  
Contract DE-AC07-05ID14517**

# Architectural and Algorithmic Requirements for a Next-Generation System Analysis Code

V.A. Mousseau and R.W. Youngblood

05/31/2010

## Abstract

This document presents high-level architectural and system requirements for a next-generation system analysis code (NGSAC) to support reactor safety decision-making by plant operators and others, especially in the context of light water reactor plant life extension. The capabilities of NGSAC will be different from those of current-generation codes, not only because computers have evolved significantly in the generations since the current paradigm was first implemented, but because the decision-making processes that need the support of next-generation codes are very different from the decision-making processes that drove the licensing and design of the current fleet of commercial nuclear power reactors. The implications of these newer decision-making processes for NGSAC requirements are discussed, and resulting top-level goals for the NGSAC are formulated. From these goals, the general architectural and system requirements for the NGSAC are derived.

## Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Overall Code Structure and Development Strategy</b>	<b>6</b>
2.1	Generic Component . . . . .	6
2.1.1	Load Equations . . . . .	7
2.1.2	Capacity Equations . . . . .	8
2.1.3	Reliability Equations . . . . .	9
2.1.4	Aleatory Life Equations . . . . .	9
2.1.5	Aleatory Clock . . . . .	9
2.1.6	Static Model Parameters . . . . .	9
2.1.7	Control Model Variables . . . . .	10
2.1.8	Input Processing . . . . .	10
2.1.9	Output Processing . . . . .	10
2.1.10	Initialization . . . . .	10

2.1.11	Numerical Methods . . . . .	11
2.2	Generic Interface . . . . .	11
2.2.1	Aleatory Uncertainty Component . . . . .	11
2.2.2	Operating Procedures Component . . . . .	12
2.2.3	Uncertainty Quantification . . . . .	12
2.2.4	Data Assimilation . . . . .	13
2.2.5	Optimization . . . . .	14
2.2.6	Control System . . . . .	14
2.2.7	Figure of Merit . . . . .	14
2.2.8	Communication with the Code User . . . . .	14
2.3	General Code Overview . . . . .	15
2.3.1	Graphical User Interface . . . . .	15
2.3.2	Database . . . . .	16
2.3.3	Validation Data . . . . .	16
2.3.4	Economics Component . . . . .	16
2.3.5	Components . . . . .	16
2.3.6	Global Physics . . . . .	17
2.3.7	Generic Capabilities . . . . .	17
2.3.8	Control . . . . .	18
2.4	Software Development Strategy . . . . .	18
2.4.1	Validation . . . . .	19
2.4.2	Software Quality Assurance . . . . .	20
2.4.3	Open Source . . . . .	21
<b>3</b>	<b>Generic Capability</b>	<b>21</b>
3.1	Homogenization . . . . .	21
3.2	Solver and Preconditioner . . . . .	22
3.3	Uncertainty Quantification . . . . .	22
3.3.1	Unobtrusive . . . . .	23
3.3.2	Obtrusive . . . . .	23
3.4	Data Assimilation . . . . .	24
3.4.1	Static . . . . .	25
3.4.2	Dynamic . . . . .	25
3.5	Optimization . . . . .	25
3.5.1	Mont-Carlo Based . . . . .	25
3.5.2	Newton Based . . . . .	26
3.5.3	Vulnerability Search . . . . .	26
3.6	Grid Generation . . . . .	26
3.7	Time Step Control . . . . .	27
3.8	Distribution Sampling . . . . .	27
3.9	Figure of Merit . . . . .	27

<b>4</b>	<b>Global Physics</b>	<b>28</b>
4.1	Equation of State . . . . .	28
4.2	Material Properties . . . . .	29
4.3	Multi-Phase Multi-Component Fluid . . . . .	29
4.4	Neutronics . . . . .	30
4.5	Geometry . . . . .	30
<b>5</b>	<b>Components</b>	<b>31</b>
5.1	Separate Components . . . . .	32
5.2	Composite Components . . . . .	33
<b>6</b>	<b>Control</b>	<b>34</b>
6.1	Control System . . . . .	34
6.2	Operating Procedures . . . . .	35
6.2.1	Start Up . . . . .	35
6.2.2	Shut Down . . . . .	35
6.2.3	Maneuvering . . . . .	36
6.2.4	Response to Initiating Event . . . . .	36
6.2.5	Human Factors . . . . .	36
6.3	Aleatory Uncertainty . . . . .	36
6.3.1	Read Reliability Equations . . . . .	36
6.3.2	Roll Dice to Choose Maximum “Lifetime” . . . . .	36
6.3.3	Set the Aleatory Life Clock . . . . .	37
<b>7</b>	<b>Graphical User Interface</b>	<b>37</b>
7.1	Input and Output . . . . .	37
7.2	Tracking Decision Points and Sequence Spawning . . . . .	38
7.3	Study Initialization . . . . .	39
7.3.1	Uncertainty Quantification . . . . .	39
7.3.2	Optimization . . . . .	39
7.3.3	Data Assimilation . . . . .	39
7.4	Simulator Mode . . . . .	39
7.5	Database . . . . .	40
7.5.1	Inputs . . . . .	40
7.5.2	Outputs . . . . .	40
7.5.3	Validation Data . . . . .	40
7.6	Surrogate Construction . . . . .	41
7.6.1	Low Fidelity . . . . .	41
7.6.2	High Fidelity . . . . .	41
7.6.3	Unused Computer Time . . . . .	42
7.7	Knowledge Management . . . . .	42
7.8	Economics Component . . . . .	42
7.9	Check if the Simulation is Physically Reasonable . . . . .	43
7.10	Experimental Design . . . . .	43
<b>8</b>	<b>User</b>	<b>43</b>

<b>9</b>	<b>Summary</b>	<b>44</b>
<b>10</b>	<b>Acknowledgment</b>	<b>46</b>
<b>A</b>	<b>Table of Requirements</b>	<b>46</b>

## List of Figures

1	The requirements for a general component. . . . .	6
2	The general code layout requirements. . . . .	15

## 1 Introduction

This software development project is based on two important concepts, “What is possible?” and “How the software will be used.” is different. What is possible in reactor safety modeling has changed significantly since the last time someone in the United States started from scratch to write a new reactor safety code (which was about 35 years ago). In the last 35 years, computer memory and speed have increased many orders of magnitude, numerical methods have exploded in their ability to provide stable, accurate, and robust solutions. Experimental instrumentation and the resulting physical understanding also have made dramatic improvements. Software engineering has matured since the 1970’s to what it is today. Additionally, how the software can and will be used has changed significantly. In the 1970’s, the Nuclear Regulatory Commission (NRC) licensing was based on analysis of selected Design Basis Accidents (DBAs), surrogates that were meant to envelope the physical challenges posed by probabilistically significant scenarios. These analyses were simplified by the conservative assumptions and only involved a few transients. Modern risk-informed approaches consider the broader set of probabilistically significant scenarios. This requires a very large number of runs and includes a much larger set of active and passive components.

Decision support requires uncertainty quantification and intelligent uncertainty reduction. Modern safety analysis recognizes that there exists two forms of uncertainties that impact computational results. The first is epistemic uncertainty, which accounts for lack of knowledge. This includes incomplete experimental data, physics left out of a model, and other uncertainties that could be reduced by having more time and more money. The second is aleatory uncertainty or unknowable uncertainty. For this document, aleatory uncertainty will be broken into two separate components. The first aleatory uncertainty will come from the simple recognition that there is some error in an experiment that cannot be removed. If the same experiment is repeated ten times, there will be ten different results. This distribution of results from the same experiment will be accounted for as an irreducible range in the model parameter. This irreducible range in the parameter results in an irreducible uncertainty that defines a practical “floor” for how much uncertainty can be reduced by experimentation.

The second aleatory uncertainty comes from purely random acts that cannot be accounted for in the computer model. This means that there is no reasonable way to predict these events, and they need to be included in the model as a stochastic event. The Risk Informed Safety Margin Characterization (RISMC) part of this project will include analysis of both the aleatory and epistemic uncertainty and how they impact the safety margin of the reactor.

Historically, due to small computer memories, slow computer speed, and primitive software engineering practices, one was provided with limited options for simulation of reactor safety performance. By definition, it was “good enough” because it was all that was possible. Today, one has many different options, including computational fluid dynamics (CFD), neutron transport, and traditional system codes. In addition, one can build hybrid models that combine these at different locations in time and space. The key is to be able to answer the question, “Which of these methods is good enough in a specific context?” Clearly one does not want to spend time computing on a more complicated model if a cheaper answer is good enough in a particular situation. Therefore, the ability to assess the uncertainty of the calculation will be designed in from the beginning. Based on the quantification of uncertainty, one can determine whether the uncertainty is low enough to simplify the decision. If not, a value-of-information assessment can be done to determine the value of uncertainty reduction: for example, the value of employing better models, higher spatial resolution, a more detailed physical model with less assumptions, or new experimental data. The code will be designed with the ability to rapidly change models in order to decrease uncertainty. To accomplish this, components will be designed that can be solved at different levels of geometry resolution (0-D, 1-D, 2-D, and 3-D) and different levels of physical assumptions (compressible or incompressible) and the ability to quickly make use of new experimental data to calibrate models to be more accurate. This automated tuning process is called data assimilation.

The final major improvement will be to address the impact of aging on nuclear reactor safety. This will be accomplished through the inclusion of equations that account for component failures. These failures may be due to aging, a violent physical transient, or even just random occurrences due to aleatory uncertainty. The solution of these equations will provide the code with the ability to predict the effects of aging on the safety margin of a nuclear reactor. In the beginning, these predictions will come with a large uncertainty. It’s possible that the uncertainty will be large enough that the equations will not yield useful quantification of the uncertainty. During this time of large uncertainty, they can still be used for analyzing trends, such as one change in the operating procedure increases the safety margin and another decreases the safety margin.

The rest of this document will have the following structure. Section 2 will contain a description of the general code structure and the process for improving the code through uncertainty reduction. Section 3 lists the generic analysis tools that will be used. Section 4 describes the physics models that are required to build the components. In Section 5, the requirements from Sections 3 and 4 will be implemented into the physical components that make up the nuclear reactor. Section 6 describes the different types of run time controls. Section 7

presents the graphical user interface. Section 8 describes the code user. Section 9 presents the reader with the summary. Appendix A lists the requirements in table form.

## 2 Overall Code Structure and Development Strategy

This section provides the requirements that are needed for component models to enable the capabilities required for the RISMC mission and this is shown in graphical form in Fig. 1.

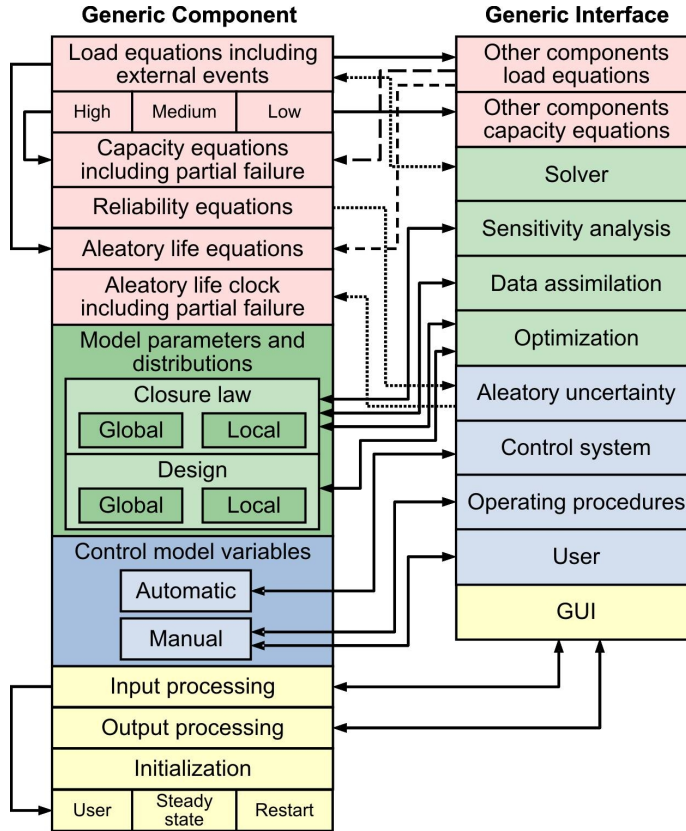


Figure 1: The requirements for a general component.

### 2.1 Generic Component

This subsection describes the pieces of a generic component assuming all code options are employed. The code structure will be put into place to enable these

different applications. It is important to note that these capabilities do not always have to be employed. If a faster code version is required, one can choose to turn-off any capability that is unwanted. By turning different capabilities on and off, one can run the code in a variety of different modes.

1. Prescriptive - this is the traditional Appendix K approach. The events are prescribed and the parameters are selected to be conservative.
2. Predictive - In this mode the event is prescribed but the code will run in Best Estimate Plus Uncertainty (BEPU) mode. Here, the best prediction is made and the uncertainty of that prediction is quantified.
3. Stochastic - Here, the event is sampled from a distribution and the results are summarized in terms of a probabilistic distribution function or a cumulative distribution function. This is the mode for RISMC.

The code user now has a large amount of flexibility on how to use the simulation tool. The different parts of the generic component will now be discussed.

### 2.1.1 Load Equations

The load equations account for the physics happening inside of the component. They will normally look like conservation laws of some type such as the following:

- Mass
- Momentum
- Energy
- Neutron number density
- Angular flux.

These are the traditional equations for simulation codes. What is different from traditional codes is that these equations are required to have multiple levels of fidelity. The requirement for multiple levels of fidelity originates from three separate requirements. The first requirement is for the purposes of uncertainty quantification. In order to quantify the uncertainty in a given model, we need to be able to assess the uncertainty of the assumptions in that model. In order to assess the uncertainty based on an assumption, we need two models that differ only by that assumption. For example, consider the evaluation of the often used assumption of incompressible flow. This also could be a 0-D model of a tank versus a 1-D model of a tank. From the neutronics side, it could be point kinetics versus 1-D neutron diffusion. The key to assessing uncertainty of the modeling assumption is to ensure that the two models give the same answer when the assumption holds. One can then start a transient where the assumption holds and compare how the solutions differ when the assumption is violated. The second reason for multiple levels of fidelity is the obvious trade-off between speed and accuracy. If lower levels of accuracy are acceptable, then simpler

models that run faster can be employed without any negative impact. The final reason for this requirement is the most important. By comparing different models that remove assumptions, one can evaluate the sensitivity of the solution to that assumption. If the assumption has a large uncertainty, then this is an area where resources should be applied to make a significant reduction in the uncertainty of the model. This provides a quantified process for determining what is the most important area in which to invest future resources.

In addition to the load on the components that come from the conservation equations, there is a second load source. This is the load that results from external events. Examples of external events are as follows:

- Fire
- Flood
- Earthquake
- Tornado
- Hurricane.

These external events will provide additional load on the components not accounted for in the conservation laws. To enable the II-group method of uncertainty quantification and to reduce computer round-off errors, all load equations need to be nondimensionalized.

### 2.1.2 Capacity Equations

The capacity equations account for the physical mechanisms that can mechanistically cause component failure. These equations include simple failure modes such as pipe rupture due to exceeding the maximum pressure. The capacity equations also have a more complicated role in computing the aging of the component. This is where thermal stress, mechanical stress, and radiation-induced damage is accumulated over the life of the component. The basic equation will look like this:

**if** *stress* > *strength* **then** *the component will fail*.

This is a significant difference from traditional reactor safety codes. First, the code itself will determine when a component fails, instead of the user determining this *a priori* and specifying it in the code input. This means that the software now has to include operating procedures to respond to a component failure. The second change is that transients behave differently when they are run at different times during the lifetime of a reactor. Because of the aging effects, a component that did not fail in the first 10 fuel cycles may now fail in the 11th fuel cycle. In this sense, the reactor will always be in some transient mode due to aging. To enable the II-group method of uncertainty quantification and to reduce computer round-off errors, all capacity equations need to be nondimensionalized.

### 2.1.3 Reliability Equations

Inclusion of reliability information has historically been done in probabilistic risk assessment (PRA) software. Direct incorporation of reliability information into a reactor safety code is new. The reliability equation for a given component will give the probability density of failure time, based on nominal reliability function parameters (e.g., failure rate, shape parameter, and time scale parameter). This is a purely statistical view of component failure. When the code is being used to quantify probabilistic load-capacity spectra, this function will be sampled initially within each time history to determine a nominal failure time. As discussed below in sections 2.1.4 and 2.1.5, within a given time history, this nominal failure time may be modified as a result of scenario phenomenology affecting the effective values of the reliability parameters.

### 2.1.4 Aleatory Life Equations

These equations provide a link between the purely physical capacity equations and the purely statistical reliability equations. These equations take a purely statistical component lifetime and modify it. If the component is exposed to a “bad” environment, then its life is shortened. If the component is exposed to a “good” environment, then its life is extended. In other words, this computes the current estimate of component lifetime, based on whether current system conditions are accelerating or decelerating the component’s aging rate. For example, if a component’s life is shortened by exposure to high temperatures, and temperature has just increased, then the current estimate of the component’s lifetime will be shortened relative to the previous estimate of the component’s lifetime. This is not the same as exceeding a hard threshold and mechanistically causing immediate component failure.

### 2.1.5 Aleatory Clock

This computes the maximum time that a component will live. This model has a fixed number of “ticks” of the aleatory clock. The ratio of an aleatory “tick” and a real second is determined by the aleatory life equations. The component fails when the aleatory clock runs out.

### 2.1.6 Static Model Parameters

Static model parameters are the parameters that are constant throughout a simulation run. All parameters will be input into the code as a range, a distribution, and a value. Other software can adjust the parameter value as long as it stays within its acceptable range. The distribution information is required for statistical sampling. Examples of static parameters are wall friction coefficients and wall heat transfer coefficients. Another example would be a design geometry number like the height of the heat exchanger relative to the core or a tank volume. These parameters are set once and left constant for a transient.

These are the types of parameters studied for uncertainty quantification or design optimization. These parameters are subdivided into two classes: local and global. The global parameters are shared by all components. An example of a global parameter would be a parameter in the equation of state (EOS). A change in the EOS will impact all components. A local parameter is one whose value only impacts a single component. An example of a local parameter would be a shaft friction coefficient in the turbine model. Changing this parameter will only impact the turbine and not any other component.

A list of global parameters will be maintained. When a global parameter is used in a component, it will be required to use the one in the global list. If component-specific physics are required, a local parameter will be created and used. Local copies of global parameters will not be allowed because this will corrupt the uncertainty quantification of the global parameter.

#### **2.1.7 Control Model Variables**

Control model variables are variables that change during the transient (e.g., the pump speed or the valve position). These variables are divided into two categories. The automatic variables are the ones that are set by the control system in the reactor. These responses require no human interaction. The second set of variables are changed manually by the operator. In this code the actions of the nuclear reactor operator will be accounted for by the operating procedure component. A second mode is for running the code in simulator mode where the code user will interact with the code through the graphical user interface (like a video game).

#### **2.1.8 Input Processing**

Input processing is the information provided by the code user to describe the component, geometry, capability, and connectivity to other components. Input processing will always be done in three dimensions. If a 0-D model or a 1-D model is required, they will be computed as simplifications of the full detailed information. This will allow for the input to remain static while the code functionality can dynamically change underneath.

#### **2.1.9 Output Processing**

Output processing is the information computed by the component that will be made available to the code user. This needs to be designed to provide important information in an easy-to-digest format.

#### **2.1.10 Initialization**

Initialization will be done from a restart file. The Graphical User Interface (GUI) will provide the interface with the code user. The interface between the GUI and the solution algorithm software will be through a restart file. This allows for a separation between the computer science-intensive GUI and

the math-intensive solver. The three options for the GUI to implement for initialization include the following:

1. User Specified - this is the most difficult because there is no assurance that the code user can pick an initial point that is a solution. If the initial state is not a solution, then there will be an initial non-physical transient.
2. Steady State - this option is less difficult but can still be problematic. The problem comes from the requirements being over specified so no solution exists or being underspecified so an infinite number of solutions exist.
3. Restart from a Previous Transient - this is the easiest and most reliable. This should be the desired mode of operation whenever it is possible.

#### **2.1.11 Numerical Methods**

The numerics of the code will be designed to be simplified easily so it can provide the required level of accuracy and the service required by the functionality of the rest of the code. The load equations will be designed so that they can collapse dimensionally and temporally as required. The component will contain three spatial dimensions in all cases. The discretization in all three dimensions also will be input. If the discretization in a given dimension is set to one, then that dimension is collapsed from the solution procedure. If all three dimensions are set to one, then the result is a 0-D component. Similarly, the component can be either transient or steady state. The equations will be written so the steady state portion and the transient partition are clearly separated.

To enable the II-group method of uncertainty quantification, the equations will be nondimensionalized. This will result in a nondimensional parameter multiplying each term in the physic equations. In addition, the code will have the ability to set these nondimensional parameters to zero to allow an easy method to simplify the equations by turning off unimportant physical terms.

## **2.2 Generic Interface**

The generic interface software serves as the traffic cop for controlling the flow of formation between different parts of the software. The generic interface also controls the information exchange between the solver and the component. This is required for the implicit nonlinear solution method which requires iterations of the physics equations. In addition, it interacts with the following list of components.

### **2.2.1 Aleatory Uncertainty Component**

The aleatory uncertainty component accounts for the randomness in component failure. The capacity equations account for the physical causes of component failure, but there is a stochastic nature to these failures that will not be captured in the capacity equations. This random component failure comes from the reliability equations. When a component is created, it sends its reliability equations

to the aleatory uncertainty component. A random selection is made from these equations to set the component's aleatory lifetime. This lifetime information is sent back to the component as the final time in the aleatory life clock. When a component fails (i.e., end of aleatory life or for a physical reason), a transient is induced. If the transient does not severely damage the reactor, the reactor will shut down according to the operating procedures component. At this time, the component is replaced and the new component is given a new aleatory life expectancy and the transient continues. The ability to fail components and replace them without human intervention is one of the key requirements for long-time transients.

### **2.2.2 Operating Procedures Component**

The operating procedures component codifies the actions of the nuclear reactor operator and the maintenance staff. It consists of the following six main tasks:

1. Starting up the reactor
2. Shutting down the reactor
3. Maneuvering the reactor
4. Responding to initiating events such as a component failure
5. Inspecting components for damage
6. Scheduling replacement of components

This component will control the manually changed control model variables in the reactor such as pump speed and control rod position. This component will be impacted by human factor modeling to determine the quality and efficiency of the human interaction in the system. This also is where operating procedures will impact reactor safety.

### **2.2.3 Uncertainty Quantification**

Uncertainty quantification is one of the main improvements in this new reactor safety code. Traditionally, uncertainty quantification is left as an afterthought and is done with some form of "third party" software. The problem with third party software is interfacing the two separate codes together. However, if one builds uncertainty quantification in from the beginning, the capability can be made easier to use and more complete because it is done from the inside of the code where all of the required information is available. When third party software is employed, only parameters available through input processing can be studied. To minimize the user effect, many parameters will not be adjustable through input. However, there may be a large number of parameters that need uncertainty quantification. An example of a parameter that needs uncertainty quantification that should not be part of the input is the critical Reynolds number. This parameter determines the transition from laminar to turbulent

flow. Although its uncertainty may be important, one would not want code users “tuning” this parameter in the code input.

The uncertainty quantification process consists of the following three steps:

1. Pick the parameter value from the parameter distribution, employing some sampling procedure
2. Run the simulation code with that parameter choice
3. Evaluate a figure of merit from the code output.

There are a variety of sampling procedures, but it is important to note that the uncertainty is being calculated with respect to a figure of merit. Correct choice of the figure of merit that best describes the plant behavior in a single number is a difficult task. Peak clad temperature is an example of a commonly used figure of merit for reactor safety.

There are two main classes of uncertainty quantification methods: obtrusive and unobtrusive. In the unobtrusive methods, the simulation code is simply considered a “black box.” Signals are sent in and the resulting signals that come out are analyzed. The unobtrusive methods are most popular because they do not require any knowledge of the simulation code. In the obtrusive methods, the simulation code needs to be modified to be able to get the uncertainty quantification information. These methods are less popular because they require more work. However, because they make use of knowledge of the code, more accurate results can be obtained with less assumptions.

It is important to note that uncertainty (sensitivity) is a function of both space and time. There are physics that are both a source of uncertainty and a sink of uncertainty. There are mildly annoying uncertainties that result in small perturbations in the solution that quickly die out. There also are uncertainties that continuously push the solution in one direction due to a bias. These uncertainties grow monotonically in time and are very dangerous in long-time transients.

Additionally, there are two main purposes for uncertainty quantification. The first is to simply place a number on a calculation to provide the code user confidence, “The answer is 27 plus or minus 3%.” This mode is the most popular use of uncertainty quantification. The second use is when uncertainty quantification is coupled to uncertainty reduction. In this mode, it is important to know the source of the uncertainty so that it can be reduced. Here, one needs an uncertainty quantification method that provides uncertainty as a function of both space and time. When space and time uncertainty information is available, a spike of uncertainty can be traced to what component is at that location and what model “turned on” at that instant in time. In uncertainty reduction mode, uncertainty quantification is an important guide to code development.

#### **2.2.4 Data Assimilation**

Data assimilation is the process of adjusting (i.e., calibrating and tuning) parameters, within their acceptable ranges to improve the code’s match to exper-

imental data. This process is required because inevitably, some small physics contribution is not included in the model and is accounted for through minor modifications of the parameters. This process should focus on the parameters with the largest uncertainties first and then work its way down to parameters with lower uncertainties. This process will adjust the parameters, within their acceptable ranges, to minimize the distance between the code calculation and the experimental data. Therefore, this part of the interface needs to be able to read the parameter data, including the range and distribution from the component, it needs to be able to read the experimental data from the validation database. It also needs to be able to write the calibrated parameters back to the component changing its setting of the parameter. This process can work with control model variables and static parameters.

### **2.2.5 Optimization**

Optimization is similar to data assimilation. Again the parameters are adjusted inside their acceptable range. However, this time the optimization is done with respect to a figure of merit (FOM), not experimental validation data. The process is to read the parameters and their distribution, then evaluate the FOM. An optimization algorithm changes the parameters until the FOM is either minimized or maximized. This search process requires the ability to change the parameters in the component's equations. This process can work with static parameters or control model variables.

### **2.2.6 Control System**

This is the model of the control system in the reactor. Like other components, the control system can fail for physical reasons (e.g., electrical fire) or for stochastic reasons (e.g., active component unreliability). The control system will read physical data from a large number of components and then adjust control model variables based on the control system logic. Note that this component does not interact with static parameters.

### **2.2.7 Figure of Merit**

The FOM is important for uncertainty quantification and for optimization. The FOM can take input from a large number of components and give back a single number that represents the status of the reactor. The uncertainty quantification and optimization will be done relative to the FOM.

### **2.2.8 Communication with the Code User**

This is the communication path between the simulation and the code user. In “batch” mode, this is a once-per-run interaction. The code user specifies the initial condition and waits for the final results. A second type of batch mode is when the user requests the code to perform a study based on an initial state. Here, many code runs may be completed and post processed before the data is

presented to the code user. Finally, there is the simulator mode. In simulator mode, the code can be used as a simulator to train reactor operators or to execute human factors experiments. In this mode, the code simulates the state of the reactor and the code user acts in the role of reactor operator. Here, input from the code user changes the results of the simulation “on-the-fly” similar to a nuclear reactor responding to the reactor operator changing the controls.

## 2.3 General Code Overview

The general code structure is shown in Fig. 2. This figure depicts the overall code structure and the interactions between the major software structures.

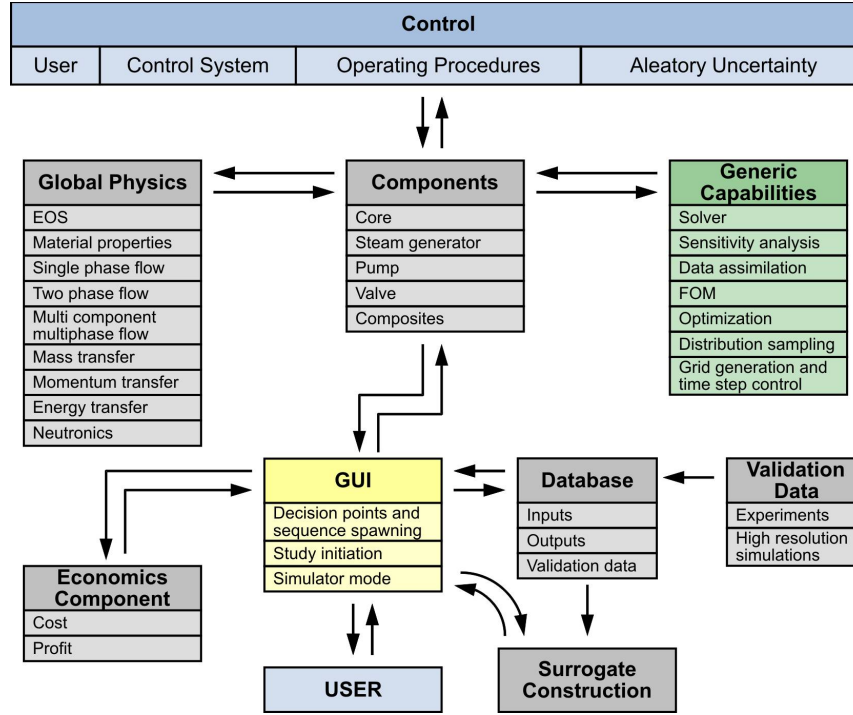


Figure 2: The general code layout requirements.

### 2.3.1 Graphical User Interface

Starting from the code user, the first piece of software is the GUI. From the code user perspective, this is the most important part of the code. The GUI is the first and most important place to enforce code ease of code use. Building of input that describes the nuclear reactor geometry and connectivity needs to be easy to use and intuitive. The initiation of a study (i.e., uncertainty quantification, data assimilation, optimization, and risk assessment) should be

straight forward and easy. The GUI provides the conduit for information from the code user to the code and from the code to the code user. In simulator mode, the GUI should respond in almost “real time” to the code user’s changes in the plant’s configuration.

### **2.3.2 Database**

A database is required to keep track of the large number of inputs and outputs from the different types of studies. This database will track the code inputs and corresponding outputs and will provide an easy way to reproduce results. In addition, the code will need a database to store the experimental data used to validate components. If the database is loaded directly as the experiment is run, this will enable a new mode of code calibration called 4-D data assimilation. The fourth dimension is time. As new data comes available, the parameters in the code can be calibrated to account for the new information. This is similar to how meteorological data are used in weather forecasting.

### **2.3.3 Validation Data**

Validation data need to be stored in the database. Whenever possible this will come from experimental data. Ideally, the data will come from an experiment that was designed specifically to reduce an important uncertainty in an important transient. Whenever real experimental data are too costly or too dangerous to acquire, high resolution simulations can be used to augment the experimental data. Well chosen high fidelity simulations may reduce uncertainty when little experimental data are available.

### **2.3.4 Economics Component**

There are many transients that do not pose a risk to the general population or the environment that will have a significant impact on the reactor. Although these transients do not pose a safety risk, they can pose significant economics risk either in causing capital equipment replacement or in forcing significant plant downtime, possibly as a result of regulatory action. To account for the economic impact of operating procedures, we will include a simple economics component. Initially, this component will incur cost when components are replaced and will generate revenue proportional to the power level when the plant is operating. The difference between revenue and cost will provide a very simple model of profit. Inclusion of an economics component allows one to perturb operating procedures and calculate the impact in terms of both economic risk and safety risk.

### **2.3.5 Components**

Moving upward from the GUI in Fig. 2 one comes to the component block. This is the heart of the engineering of the simulation. This is where all physics and engineering models that are specific to the component are located. As the

number of nuclear reactors to be analyzed increases and as the scope of transients on those reactors to be analyzed increases, the number of components will grow rapidly. Therefore, it is necessary to accelerate the development and deployment of large numbers of quality components. The goal is to allow multiple code developers to be able to document, build, test, and verify and validate components in an independent manner. This requirement of being able to scale up the number of code developers is vital to the success of this project.

Another important requirement in the code design is the ability to build composite components. The idea here is to combine basic components (e.g., pipes, heat conduction, and point kinetics) to create a composite component such as a “core.” The ability to design, test, and verify and validate composite components could greatly increase the speed and quality of building new reactor input descriptions.

### 2.3.6 Global Physics

To the left of the components block in Fig.2 is the global physics block. It is important to distinguish global physics from local physics. Local physics are physics that only impacts the current component or composite component (e.g., the conversion of momentum to energy in a turbine). This physical process is not required in other components. A good example of global physics is the EOS. The EOS will be the same for all of the components using that working fluid. The person writing a pump component will not have his own EOS, he will use the global EOS.

Implementing global physics requires having subject matter experts (“domain” experts) on the code development team: the knowledge required to combine neutronics, thermal hydraulics, and risk is too much for one person to be expected to possess. Therefore, we need to minimize the expertise needed in the component building process. For example, it makes sense for the project to have a heat transfer expert; this person would work on global heat transfer solutions that can be used by any component. If it is discovered that the global heat transfer solution has too large an uncertainty for a given component, such as a pump, then the heat transfer expert can advise the pump component author on how to proceed. This approach provides subject experts that are available to aid in the component design and testing. This minimizes the knowledge required by component authors.

### 2.3.7 Generic Capabilities

To the right of the components block in Fig. 2, is the generic capabilities block. This block describes the capabilities that are required to perform an analysis. The analysis is independent of the reactor or the transient because that information is encapsulated in the components. An example of a generic capability is the ability to solve the component load equations. The solution of the component equations will be done with a physics-based preconditioned, Jacobian-free, Newton-Krylov (JFNK) method. The generic capabilities interaction with

the component are mainly of the form of sending a state vector and evaluating the component equations. The sensitivity analysis, data assimilation, and optimization capabilities require the ability to change parameters within their distribution range. This requires the parameters to have a distribution that the generic capabilities can sample.

It is important to note that there are both global parameters that may be perturbed for all components and local parameters that only impact the current component. Ensuring that the components use the global parameters and do not make local copies is an important requirement to enforce. The analysis of the sensitivity to a global parameter needs to be felt throughout all components.

Finally, the two hardest and most important generic capabilities are time step control and mesh generation. One of the largest sources of the “user effect” (two different code users getting two different answers to the same problem) in current reactor safety codes is the current practice of the time step and the computational mesh being set by the code user. There are two important benefits from automating these processes. First, if the grids and the time steps are the same for two different users, then there is a higher probability that they will both get the same answer. Second, one of the main components of code verification is mesh and time step convergence studies. If this process is automated, then these convergence studies can be done with very little human intervention required.

### **2.3.8 Control**

Control is the part of the software that sets the model control variables. There are three types of control. The first is the control system that models the real control system in the reactor. This is where valves are opened and closed based on design set points. The second is the operating procedure component. This accounts for the actions of the reactor operator (or his surrogate - the code user in simulator mode). The final part is the aleatory uncertainty component (stochastic component failures). This accounts for the randomness in the component failures. All of these controls change the state of the component through its automatic and manual control model variables.

## **2.4 Software Development Strategy**

The software development strategy will roughly follow the outline below:

1. Choose a reactor to analyze.
2. Choose a family of scenarios to study corresponding to a PRA event tree.
3. Choose a subset of the event tree sequences to study.
4. Choose a list of generic capabilities to perform the study.
5. Choose the global physics required to study the transient.

6. Create a list of components required by the reactor, the transient, and the PRA event sub-tree to be studied.
7. From items 4 and 5, one can create a detailed requirements document for the components listed in item 6.
8. In an overlapping phased approach, first complete the work on item 4, then 5, and finally 7.
9. Set decision points to test progress and readjust the lists in 4, 5, and 6 to be shorter or longer, depending on whether the project is going slower or faster than planned.
10. Verify and validate the components and document the completed study.

The code then evolves by adding new reactors and new transients to be studied. Additionally, the code evolves by adding generic capabilities and more global physics (e.g., noncondensable gases and boron). Finally, the list of components also will increase and the code capability will become more complete. As the component list, generic capabilities, and global physics are increased, it will take less effort for each new reactor and new transient to be studied, because most of the work will be already completed. Three key processes in the software development strategy are described in the following text.

#### **2.4.1 Validation**

One of the most important components of building confidence in a nuclear reactor safety code is the experimental validation. This process needs to be tightly coordinated with the code development to get the optimal value from the costly and slow experimental process. The following key points will accelerate this process:

1. Simulate the experiment before, during, and after the experiment. The code should be used to design the experiment and help determine how to run the experiment, what to measure, and where to measure it. If possible, 4-D data assimilation should be used while the experiment is running to ensure that enough data have been gathered. Once the experiment is done, the code needs to be run to compute the reduction in uncertainty that resulted from the experiment.
2. Experiments need to be designed to reduce uncertainty in risk-important transients. A significant amount of simulation work needs to be done to choose a high uncertainty that needs to be reduced with experimentation. In addition, placement of the experiment in state space can be optimized to have the largest impact on uncertainty reduction. Similarly, how the experiment is run and what is measured will be optimized before the experiment is performed.
3. Experiments should be optimized to quantify the following

- (a) Physical Response Time Scales - perturb the system and determine how long it takes for the system to respond.
  - (b) Physical Response Length Scales - perturb the system and determine how far away the impact of the perturbation reaches.
  - (c) Multi-physics Coupling - include multiple physics and the constraints and feedback mechanisms in the experiments.
  - (d) Experimental Sensitivity - perturb the boundary conditions and measure the experiment's response. The experiment's sensitivity then can be compared to the code's sensitivity.
  - (e) Aleatory Uncertainty - repeat the same experiment to provide a lower bound on the uncertainty reduction possible from the experiment. This comes from the distribution of outputs from identical experiments.
4. The parameters and the physics are divided into two categories: local and global. Values of local parameters contained in a component should be initialized based on Separate Effects Tests (SET). Values of global parameters should be initialized based on Integral Effects Tests (IET). Therefore, one should first set the local parameters based on SET; then one should set the global parameters based on IET. IET should only be used to modify local parameters after they have been calibrated with SET.

#### **2.4.2 Software Quality Assurance**

Software quality assurance (SQA) is one of the processes that build confidence in the software. SQA will be done based on the following five documents:

1. Requirements Document - this describes the capabilities of the software.
2. Testing Plan - this describes how one will demonstrate that the required capabilities have been delivered. Tests and answers are described in detail. This includes both verification and validation.
3. Software Design and Implementation Document - this describes the structure of the software and the subroutines and variables. This is the information that is needed to maintain the software.
4. Acceptance Testing - this is where the testing plan is executed on the completed software. The computed answers should match the predicted answers. Any discrepancy between the expected results and the actual results needs to be described in detail.
5. Developmental Assessment - this document demonstrates that the code can be used for its designed purpose with confidence.

Steps 1 through 4 will be done for individual sections of software (e.g., the solver, the pump component, or the two-phase flow equations). The last step, developmental assessment, is a global measure of the code's capability. Developmental

assessment is performed for the whole code after a large capability change has been implemented.

The purpose of SQA is to provide confidence in the software and to make code maintenance more productive. Drafts of the SQA documents should be done along the way and pilot software should be written. The final version of the SQA documents and the production version of the software should not be done until the decision points built into the plan have passed (where capabilities are added and subtracted). Therefore, detailed SQA should be left to the final phase of the development when production software is being produced.

### **2.4.3 Open Source**

This project needs to be able to recruit a large number of code developers for the labor-intensive job of building, documenting, verifying, and validating the large number of components required for this project. There are numerous distinct designs in the current commercial reactor fleet in the United States, comprising distinct systems and components types. Each of these reactors have hundreds of components, and the probabilistic margin assessment mode requires analyzing many scenarios. The resulting number of components required to address life-extension decisions is very large. If this software is export controlled, proprietary, or sensitive, the number of people that can be allowed to contribute is greatly reduced. Therefore, it is very important to keep most of the software open source so that all university students and professors have access. The small portion of this software that is export controlled, proprietary, or sensitive will be handled separately.

## **3 Generic Capability**

This section discusses the reactor generic capabilities that are required to support RISMC. In the following nine subsections (3.1 - 3.9) different generic capabilities will be discussed.

### **3.1 Homogenization**

The key to having multiple levels of fidelity in a component is the ability to homogenize information from a high fidelity model down to a low fidelity model. Homogenization removes the unimportant high frequency information, but maintains their important impacts on the solution through closure laws. This work is mature in the field of neutronics where this process has been a key aspect of daily business for years. Although this type of work is common in neutronics, it is seldom employed in thermal hydraulics. Determining how this issue will be addressed in thermal hydraulics is an important early research project. The ability to homogenize a solution is a simple concept and does not require research. The ability to do it automatically in a manner that preserves the FOM accurately is still the subject of cutting edge research and is a stretch goal within this project.

### 3.2 Solver and Preconditioner

The JFNK technology is relatively mature. It can be acquired as software packages from Argonne, Sandia, Lawrence Livermore, or many other places. All of these packages have parallel versions of the solver available. Serial versions of this technology are very simple and can be built “in house” with little cost. The efficiency of the solver depends on the preconditioner. All of the solver packages come with their own preconditioners included, so preconditioning is a solved problem as well. The path that seems most productive in development of preconditioners “in house” is an idea called physics-based preconditioning. Simply put, one uses an existing solution method as a preconditioner for JFNK. This physics-based preconditioning approach has been shown to be a very powerful method in other fields and has great potential in reactor safety.

### 3.3 Uncertainty Quantification

Uncertainty quantification is the key to determining if the information produced in the simulation is “good enough.” The following four different uncertainties need to be quantified:

1. Parameter Uncertainty - this is the most studied and best understood uncertainty. The weakness in existing approaches are as follows:
  - (a) Not having a complete list of parameters because they are not available through input processing
  - (b) Not having range and distribution information.
2. Numerical Uncertainty - this is the uncertainty that is normally addressed separately in verification. Note that this is related to solution verification, not code verification.
3. Partial Differential Equation (PDE) Model Equation Uncertainty - There is an uncertainty associated with the accuracy and applicability of the PDE model. This uncertainty is normally not even considered. For example, if one employs a 1-D system code approach, which is standard, one has assumed that nothing important happens in the two other directions. To address the uncertainty of this assumption, one needs a 3-D solution and the difference between the 3-D solution and the 1-D solution quantifies the uncertainty in the 1-D model assumption.
4. Closure Model Uncertainty - There are times when more than one closure model may be appropriate for a given transient. This uncertainty relates to the difference in the solution when different applicable closure models are used.

There are a variety of uncertainty quantification approaches available. They vary in their maturity and their acceptance in the field. Initially, these will be employed at a low level and the downselect between different methods will be

delayed until later. The following subsection discusses the six methods currently under review that are divided into two classes: obtrusive and unobtrusive. The ability to incorporate these methods will be designed into the code from the beginning. Which uncertainty quantification method is employed will depend on resources and requirements.

### **3.3.1 Unobtrusive**

Unobtrusive methods are the only real options available for legacy codes. Because of the age and structure of legacy codes, it is difficult to “back fit” numerical capabilities. Therefore, obtrusive methods are difficult or impossible to implement in legacy software. The following is a list of unobtrusive uncertainty quantification methods being considered.

1. Code Scaling, Applicability, and Uncertainty (CSAU) - This is the oldest approach in the nuclear reactor safety area. The CSAU methodology is accepted by NRC for doing BEPU computations. This method is very expensive in terms of man power, but is a very accurate method if good subject matter experts are available.
2. Black Box - Black box methods are gaining popularity. The idea here is to use many code runs and simply determine uncertainty numerically. However, it is very important to understand the assumptions of these black box methods before employing them. First, they may assume that there is no numerical, PDE model, or closure model uncertainty (only parameter uncertainty exists). We seldom understand the physics well enough that there is no PDE or closure model error. Second, they may assume that the code user knows all of the important parameters and their distributions. Except for the case when high quality expert opinions are available, this second assumption is seldom true.
3. Dynamic Response Based - This method is based off work on dynamic PRA. In dynamic PRA, the event tree is set, but the timing of the branching in the event tree is a new parameter. One can then calculate the sensitivity of the core damage frequency due to the timing of a specific event. This uncertainty information illuminates which events require detailed timing studies.

### **3.3.2 Obtrusive**

Obtrusive methods are only possible in brand new software. Because we are writing new software from “scratch,” we can build-in the obtrusive methods from the beginning. The following is a list of obtrusive uncertainty quantification methods under consideration.

1. Forward Sensitivity Analysis (FSA) - FSA is a newer approach that is mainly still in use only in applied mathematics. The idea here is to write a nonlinear equation for the parameter sensitivity to solve for sensitivity

directly. This is an approach available to this nuclear reactor safety code because it employs a nonlinear solver capability. This method works well for a large number of FOMs and a small number of parameters.

2. Adjoint Based - This method solves for uncertainty by creating a new math problem that needs solving. From the solution of the new math problem, very detailed uncertainty information is available. This method is optimized for a large number of parameters and a small number of figures of merit. This method is very mature in the fields of linear, steady-state problems like neutronics, but is not often employed in nonlinear transient problems like nuclear reactor safety.
3. II-Group Based - This method does a nondimensional analysis of the equations. In the nondimensional analysis, each term in the equations will have a different nondimensional number. These nondimensional numbers can be compared to determine which physical models are the most important during a transient.

### 3.4 Data Assimilation

One of the “black eyes” that legacy code carry with them is the idea that the codes match data well simply because they are “tuned” to get the correct answer. This “tuning” process makes the codes look very well in terms of matching validation experiments but may have a negative impact on their ability to predict future results. This negative impact is caused by the cancellation of errors or simply put, “Getting the right answer for the wrong reasons.” This means one can match an error in the numerical method to an error in the physical model. If these two errors have opposite signs they cancel each other out. The errors have not been minimized, they have been left large and “tuned” to get the correct answer.

This approach will be improved in this software development with the following process. First one will obtain separate effects data with quantified experimental uncertainty (if none is available it will have to be created). The first step is to perform space and time convergence studies to minimize numerical errors at the point in state space where the experimental data exists. Now that numerical errors have been minimized, the next step is to minimize modeling errors. All of the different models will be compared (with space and time converged solutions) and the model closest to the data will be chosen. When the model and numerical errors are minimized, the final step is to “tune” the model parameters to best match the experimental data within the level of uncertainty of the data and the uncertainty of the code calculation. All models need to have their local parameters tuned with separate effects test data.

Then one needs to get integral effects test data with quantified experimental uncertainty. Integral effects data are used to tune global parameters so the data and the code match. It is important to note that integral effects test data are used for global parameter “tuning” and for multi-physics coupling term “tuning” and only after the local parameters are set and the model and numerical

errors have been minimized. The “tuning” process will be done in an automated process. This automation of the “tuning” process will help to minimize the user effect. There are two types of data assimilation that are being considered, static and dynamic.

#### **3.4.1 Static**

Static mode is the traditional method of using validation data to “tune” the models. We will automate this traditional approach, which is human time intensive. Based off the validation data, the parameters will be modified (within their acceptable range) until the codes matches the data. This automated approach will take into account the quality of the validation data. Each time a new set of data is available, this automated process will be run to account for the new information.

#### **3.4.2 Dynamic**

The dynamic mode is new to the nuclear reactor safety field but is well know in meteorology. In this approach, the code is run simultaneously with the experiment. Data from the experiments are collected directly from the instruments and fed immediately into the code. When the experiment is perturbed (e.g., by changing the pump speed) the code’s pump speed also is changed. Now the dynamic response of the code is compared directly to the dynamic response of the experiment. These perturbations are continued until the code has been trained to respond to the perturbation correctly.

### **3.5 Optimization**

The optimization process modifies a set of parameters to minimize (maximize) a FOM. A good example is the tuning of the input to match the reactor design set points. Certain parameters (e.g., loss coefficients) are very difficult to set and have a large parameter uncertainty range. These parameters are then adjusted to make the code output match the design set points. The FOM is the distance between the design set points and the code output. A numerical algorithm minimizes this FOM. Three types of optimization will be considered.

#### **3.5.1 Mont-Carlo Based**

The Monti-Carlo based approach is similar to the black box based uncertainty quantification method. A large number of code runs is performed and the optimal solution is determined to be the smallest FOM measured. This approach does not require any information about the code and is very robust. Its main draw back is that it can be computationally intensive.

### 3.5.2 Newton Based

In this approach, we apply Newton’s method (from the JFNK solution method) directly to the FOM. The FOM is a function of a set of parameters and Newton’s method minimizes the FOM. This approach requires intimate detail about the equations. With this knowledge, a much more efficient optimization can be obtained in a small number of code runs.

### 3.5.3 Vulnerability Search

The most interesting type of optimization is to maximize risk. In this approach, the parameters are adjusted to force the largest risk (smallest safety margin). The set of parameters that produce the largest risk is the most dangerous part of state space. Once this dangerous part of state space has been identified, operating procedures or the reactor design can be modified to remove the vulnerability. Once the vulnerability has been identified, development of a safety case proceeds in a straight forward manner. We know where the dangerous parts of state space are located, and we clearly demonstrate that our design and operating procedures provide a safe margin.

## 3.6 Grid Generation

Grid generation determines the spatial accuracy of the simulation code. Each of the multi-physics included in the reactor safety code (e.g., thermal hydraulics, neutronics, and thermal conduction) contain physical length scales. It is important to recall that these length scales change as a function of space and time. Three basic approaches for multi-physics grid generation are as follows:

1. One figures out the smallest length scale of all of the physics in the simulation. This sets the maximum grid spacing. The grid is computed to always resolve all of the physical length scales.
2. One uses a different grid for each physics so the grid only needs to resolve the length scales of a single physics. Coupling between the multi-physics needs to be addressed by mapping the physics between different grids.
3. Instead of using the same grid spacing everywhere, the grid is refined only in the regions where physical length scales are small. In this adaptive mesh refinement approach, the grid evolves in space and time with the transient.

These three approaches have been used with varying success in other fields. However, another approach exists where the equations are modified to change the physical length scales. In this approach, the equations are averaged over a length scale so high frequency information is removed from the PDE model and its effects are included instead as a closure model. One chooses the minimum length scale that is important in the physics. The equations are then homogenized to remove any physics smaller than the chosen minimum. So if

the physical length scale is chosen to be “L” (the smallest physical length scale in the equations is “L”). Grid generation becomes simple by choosing the grid spacing to be “L/5” and all physical length scales will be resolved.

### 3.7 Time Step Control

Time step control is very similar to grid generation. The straight forward approach is to simply calculate the smallest physical time scale of all of the physics and then make the time step smaller. Similar to the multiple grid approach in grid generation, different time steps can be run for different physics. The coupling between the multi-physics then becomes an issue of mapping between the different time levels of the different physics.

What is important to recall here is that the grid and time step do not have to be chosen to make the numerical error zero. The grid and the time step simply have to be chosen so that the uncertainty due to the numerical method is smaller than the parameter and model uncertainty.

### 3.8 Distribution Sampling

One begins to see parallels between uncertainty quantification, data assimilation, and optimization. They all consist of the following three steps:

1. Make a list of parameters and know the distributions of the parameters.
2. Have strategies to sample from these distributions and then run the code.
3. Be able to evaluate a FOM based on the parameter sampled.

There are a variety of sampling strategies to improve the efficiency of black box searches of state space. A few of these will be built into the code to enable the black box capabilities. In Newton’s method, one uses knowledge of the problem to determine derivative information that indicates the downward direction in the FOM. The parameters are chosen from their distribution, such that the maximum decrease in the FOM occurs. This knowledge of the derivative information is what makes Newton’s method the optimal minimization approach.

### 3.9 Figure of Merit

One of the most important problems in reactor safety analysis is the choice of the FOM. The difficulty is to be able to choose a single number that represents the state of a nuclear reactor. Once a FOM is chosen, the rest of the work is simply a large, difficult, math problem. One advantage of our field is the fact that the NRC has preselected a group of FOMs that we are all comfortable with. There are two classes of FOMs that will be discussed: surrogate based and functional based. Peak Clad Temperature (PCT) is an example of a surrogate-based FOM. The reactor is safe if the PCT is less then 2200 degrees Fahrenheit. This however is not the real concern. There are two functional concerns:

1. There is no unacceptably large fission product release.
2. The fuel stays in a coolable geometry.

Therefore, one can use the surrogate PCT or one could use the functional definitions that address fission product release and coolable geometry. These functional-based FOMs may provide a much larger margin than the surrogate PCT. The far worse case is when the surrogate is non-conservative and the surrogate test passes when the functional test would have failed.

## 4 Global Physics

This section describes the global physics capabilities of the code. The global physics will be used in all components where it is appropriate. The local physics, which only impacts the individual component, overrides global physics. Note global physics are controlled by global parameters. Perturbations in these global parameters will impact all components. Component-specific physics will be controlled by local parameters and perturbations in these parameters will not have any direct impact on the rest of the code.

It should be noted that the physics requirements are very specific to the application chosen. Is there choked flow? Is it two-phase or multi-component? How important is boron concentration to the transient? Do noncondensable gasses contribute significantly to the uncertainty of the transient? Is rapid or slow burning of the noncondensable gasses important? How sensitive is the solution to the slight motions of the solids due to fluid solid interactions? Is the solid buildup from CRUD important to the neutronics or fluid flow?

These types of questions can only be answered after a specific transient has been defined. Instead of trying to create a super-set of all possible physics, the physics will be added to the software as the applications require. This section will simply serve as a place to list concerns that will need to be addressed in detail in the software requirements documents of the specific components. In these detailed requirements documents, all physics assumptions will be listed. These assumptions will then be supported by analysis or expert opinion. In the following five subsections (4.1 - 4.5) different global physics considerations will be presented.

### 4.1 Equation of State

One of the key requirements of this software is that all components that exchange fluid need to use the same EOS. As an example, this is not true for when a CFD code and the CONTAIN code are coupled to RELAP5. This initially sounds simple if one only considers single-phase flow, but once you allow for two-phase flow with mass transfer, this becomes significantly more complicated. For example, the mass transfer rate should not change significantly simply because the fluid moved from one component to another. Enforcing this consistency between components will require significant work.

The problem becomes much more complicated when one adds noncondensable gasses (e.g., air and nitrogen) and solids (e.g., boron and CRUD). Now the noncondensable gas can be in gas form or liquid form as a dissolved gas. The solid can be in solid form or in liquid form as a solid dissolved in the liquid. One consistent EOS that accounts for solids, liquids, and gasses of multiple components and the mass transfer and energy transfer that occurs during the transitions between these phases will be difficult.

One of the hardest problems in multi-phase multi-component equations of state is the ability to compute the speed of sound in the fluid. Very often the flow out a pipe break or out of a pressure relief valve will be choked. This means that the fluid velocity exactly equals the fluid sound speed at the break or valve. Calculating the coolant mass loss rate depends heavily on getting the speed of sound in the fluid correct. This is not a solved problem in multi-phase multi-component fluids, therefore this will continue to be a large source of uncertainty.

## 4.2 Material Properties

The requirements on material properties also are significantly more complicated when one takes a holistic view of reactor safety. For consistency, we need one set of material properties that will be used for neutronics, thermal hydraulics, fuels, and strength of material calculations used to predict component failure. In addition, now that one considers aging as an important time scale to capture with the simulation code, the material properties need to be both a function of time and other physics. For example, the strength of the material may change due to neutron bombardment. The thermal conductivity may change due to the production of fission product gas. Similarly the strength of the pipe wall may decrease due to corrosion. The cross sections used for neutronics may change due to CRUD buildup on the surface of the fuel pins. All of these material degradation processes could become important in a long time transient.

## 4.3 Multi-Phase Multi-Component Fluid

One of the key assumptions in the physics of a component is the number of phases and the number of chemical components that make up the fluid (and possibly the solid). However, there is a strategy that can be employed that makes verification and uncertainty quantification simpler in these flows. It is important to recognize that one can move through a tree of equations by simply zeroing out terms and combining equations. For example, the traditional two-phase flow of thermal hydraulics is just multi-phase multi-component flow with the number of components set to one and the number of phases set to two. If the multi-phase multi-component equations are coded initially, these equations can be extensively verified to prove that they are coded correctly. Now no new verification is required for two-phase flow because one simply changes the number of parameters. In two-phase flow, there are seven equations models and three equations models. The three equation models are simply a subset

of the seven equation models. Therefore, if the seven equation model is coded and verified, then the three equation model can be employed without any new coding or new verification by simply changing parameters in the seven equation model. Similarly, the single-phase equations (of which the three equation two-phase flow is a super-set) can be simplified by assuming incompressible flow or Darcy’s law, which are simply zeroing out terms in the compressible single-phase equations. Therefore, by coding the multi-phase multi-component equations in a generalized form, one can have a code with an entire family of fluid flow solutions.

From an uncertainty quantification point of view, each of these simplifications from the most complex fluid model to the simplest fluid model is an assumption that has uncertainty. If all of these models exist in a single piece of software, then the uncertainty of each one of these simplifying assumptions can be quantified by simply turning the assumption on and off and comparing the solution.

#### 4.4 Neutronics

The neutronics equations also can be constructed in a manner that is similar to what was done for the fluid flow equations. For this software, the top of the neutronics “ladder” will be a quasi-diffusion solution. In the quasi-diffusion method, the diffusion coefficient scalar is replaced with a diffusion coefficient tensor. This 3 X 3 tensor will allow for a different diffusion rate in all three directions. This diffusion tensor comes from a transport solution to the Boltzmann equation for neutrons. This transport solution will come from software external to this package. If the off diagonal elements of the tensor are small and the diagonal elements are close to each other, one can make a isotropic assumption and replace the tensor with a scalar and get back to the diffusion equation. If one assumes that the flux shape is constant and only the amplitude varies in time, the result is the point kinetic equations. One could have stepped down from 3-D diffusion to 2-D diffusion to 1-D diffusion in a similar manner; one dimension is assumed constant and is integrated out of the equations.

The hardest part of solving the neutronics equation is getting the cross sections. The process of building the cross sections from detailed geometry information and using a high-resolution transport equation is outside of the scope of this project. However, it is important for the success of this project to ensure that there is a user friendly method to generate the required cross sections. Because of the long-time scales analyzed with this software, these cross sections also have to include the effects of fuel burn up. For example, the fission cross section should get smaller as fuel is consumed. This will allow calculations that span multiple fuel cycles.

#### 4.5 Geometry

To allow multiple levels of fidelity in the code, a key requirement is that the input describing the geometry must be done in three-dimensions. This way the

input is the same for a 3-D calculation or a 0-D calculation. Similar to the discussion above for neutronics, all of the other physics can be collapsed in a similar fashion. To build input for a 2-D model, one of the dimensions is considered unimportant (an assumption whose uncertainty needs to be quantified). This dimension is integrated numerically to provide a 2-D description of the domain. This collapsing of dimensionality is important for both run time speed and uncertainty quantification.

## 5 Components

Again similar to the physics section, the components requirements will be very specific to a given application. High-level requirements will be described here and detailed requirements will be given in the SQA documents for the production components. These components will be built so they have the capabilities from Section 3 and the physics from Section 4. Components do not interact with each other directly. All interactions with the component are through the components interface described in Subsection 2.2. The overall structure of the component is described in Subsection 2.1. The reason for this component structure is to allow components to be developed independent of the rest of the software. The person writing the component only needs to know what to expect for inputs and what to provide for outputs. In this manner, components can be written, documented, verified and validated, and software quality assured independent of everything except the interface.

The philosophy for components will be almost the opposite as that for the physics. In the physics, we created very general equations that could be simplified or expanded. This allowed a single set of equations that have a broad use. For components, the philosophy will be to only include the minimum requirements. This change in requirements comes from the need to provide SQA documentation and to verify and validate the component. Because of this, any physics local to the component is only done locally. For example, in traditional reactor safety codes, the equations are written for a variable area pipe (the equation set used to describe rocket nozzles). However, most pipes in a reactor are constant area pipes. Moreover, since the grid spacing is usually large, the variable area equations do not correctly capture the physics. The result is to carry along a lot of equations that are not used and are applied wrong when they are used.

In this software, components will be defined at a minimum requirement level. Therefore, we will have a straight pipe component with three separate sub classes: horizontal, vertical, and angled. The closure laws are different for vertical pipes than horizontal pipes; therefore there will be two different types of pipes and their closure laws will be separate. Horizontal and vertical pipes account for the vast majority of pipes in a reactor; therefore angle pipes are most likely going to be an exception. It is best to treat these exceptions on a case-by-case basis because generic closure laws for angled pipes do not exist.

This section is divided into two main subsections: separate components and

composite components. Composite components are created by combining separate components. In this manner, larger more complicated components can be created. Composite components create two types of interfaces. Local interfaces between components inside of a composite component and global (generic) interfaces that communicate with other components. The local interfaces can be made very specialized to handle their specific local job and do not need to be as generic as global interfaces. Each composite component then speaks to the rest of the components through a normal global interface. This will allow composite components to be verified and validated and software quality ensured once as a group. This will save time in the SQA process because large portions of the SQA work will already be done.

## 5.1 Separate Components

Examples of separate components are as follows:

1. Constant area and direction pipes
  - (a) Horizontal
  - (b) Vertical
  - (c) Angled
2. Variable area and direction pipes
3. Valves
  - (a) Check
  - (b) pressure relief valve
4. Pumps
5. Turbines.

The defining property of a component is the way in which it communicates with the rest of the system. For example, a pump has an inflow and an outflow. The details of what is happening in the interior of the pump is unimportant to the rest of the system. The pump can be a “black box” to all of the other components. As long as it gets the correct out flow conditions given the correct inflow conditions, nothing else matters. To this end, the pump is defined by its inputs and outputs. The pump needs to know the flow conditions coming in and whether or not the pump speed has been adjusted by a change to the control model variable (manual or automatic). The capacity equations in the pump may depend on other variables (like the water depth in the room the pump is in), but it can calculate independently when it fails. When it fails, it needs to notify the operating procedure component that it has failed.

Besides for the interface with the system, the pump model can be as simple (a constant pump head) or as complicated (a 3-D model with moving impellers)

as is needed. The simple pump is only replaced when a high complexity model reduces uncertainty or there are new requirements. In this manner the interface for the pump is designed once and then multiple pump models can be “plugged in.” All of the pump models are interchangeable and the uncertainty of different model’s assumptions can be quantified by running two different pump models and comparing the results. This object-oriented approach is the most important requirement of the software. This requirement allows multiple code developers to develop multiple pump models without ever getting in each others way.

SQA also is improved with this design. The testing of a component can be done completely independent of all of the rest of the components. The only thing required to test the pump is the pump component, its inflow interface, an inflow interface driver, the outflow interface, and an outflow interface driver. The pump component developer is provided the inflow and outflow interfaces so they only need to write the pump component and the testing drivers.

The pump component will come with a requirements document. The inflow and outflow drivers then need to test all of the pump model’s requirements and demonstrate that the pump gives the correct output given the correct input. The test plan will describe the inputs and outputs of the driver components and the pump component. Once the pump and its driver components are completed, the test plan will be implemented. If the expected results are produced, the pump and its documentation will be deemed acceptable quality. The last step is to compare the pump performance with experimental data and validate that the pump behaves in a physically realistic manner. All of this work is independent of anyone else working on the code.

## 5.2 Composite Components

At some point it will become obvious that a group of components should be connected together to create a composite component.

Examples of composite components are as follows:

1. Reactor vessel
  - (a) Neutronics
  - (b) Heat conduction
  - (c) Downcomer
  - (d) Lower plenum
  - (e) Upper head
  - (f) Pipes
2. Steam generator
  - (a) Heat conduction
  - (b) Steam dome
  - (c) Pipe

- (d) Inlet plenum
- (e) Outlet plenum.

What is obvious from these two composite components is although the intra-component connectivity is complicated, the inter-component connectivity is simple. The vessel is only connected to the hot legs and cold legs (three each for our first three loop Westinghouse PWR test problem). The steam generator is only connected to one hot leg, one cold leg, one feedwater pipe, and one main steam line. We can take a set of components that have a complicated connectivity and build a composite component that has a simple connectivity. One can follow this example and continue to build a primary composite component and a secondary composite component.

From the SQA point of view, once one has verified that the connectivity inside of the reactor vessel was done correctly, the SQA documents from the neutronics, heat conduction, downcomer, lower plenum, upper head, and pipes provide most of the SQA information for the new reactor vessel composite component. The code development team needs to define an interface for the reactor vessel component, but once that is done, the same testing procedure applies to the composite component. The final step is to validate the composite component. In this process, one can build a set of documented and tested components for the primary and secondary for the main plant designs currently in use in the United States. One would only need to SQA the changes made to the large composite components. This process will rapidly speed up the process of building input and documenting the quality of the model. One can graphically combine components and composite components that have already been SQA'ed and quickly build quality models.

## 6 Control

This section discusses the part of the simulation that does not come from the conservation laws. The control system mainly modifies the boundary conditions of the reactor. Examples are opening a valve, SCRAMing the reactor, or breaking a pipe. These types of boundary condition changes will be broken up into three sections: control system, operating procedures, and aleatory uncertainty.

### 6.1 Control System

The model of the control system includes automatic SCRAMs, pressure valves opening, and a variety of trips and set points and are the functions that exist in the control system in RELAP5. There will be a slight modification to the algebraic-based equations in RELAP5. These equations tend to be very discrete and lend themselves to computer round-off error problems.

For example, instead of saying that the valve opens when the pressure is greater than a million Pascals, the new control system would look more like the valve opens when the pressure is 10 Pascals greater than a million Pascals for

more than a tenth of a second. What this does is eliminate computer round-off error problems. During one transient, you may be “slightly” less than one million and the valve will not open. “Slightly” is related to a single bit of the 64-bit word (called machine epsilon). This sort of sharp switching can cause significant changes in the physics due to different valve opening times. This bifurcation in the system is not related to physics, but is simply a numerical artifact. By including smooth (or fuzzy logic) switching criteria, the physics is required to be above the setpoint for a short amount of time. This prevents physics from being triggered on a single computer bit. This also is a better model of the real physical instrument, which has time scales and tolerances.

The second main requirement is to input these set points as a distribution instead of a single variable; this will allow for uncertainty quantification and optimization of these parameters. Finally, the control system will need to be implicitly coupled to the simulation code. This allows one to run large time steps and still account for the control system changes. This implicit coupling of the control system will require a new type of time step control logic to be implemented. For time step control purposes, there will have to be an indicator that the control system is about to trip. This trip precursor will then cause the time step to get smaller so that the correct time of the trip is computed.

## **6.2 Operating Procedures**

This is the component that represents the behavior of the reactor operators and how they impact the system. We account for what a perfect robot operator would do (following the operating procedure exactly) and begin to account for what a human operator might do by adding in human factors to the operating procedures. For long transients, this component accounts for how plant maintenance and inspections are done. The following is a list of the main sections of the operating procedures.

### **6.2.1 Start Up**

The operating procedures will account for how the reactor starts up from a normal shut down. This start up transient can be a high risk area of simulation that have not been dealt with before. For example, this will restart the reactor after a refueling shutdown. This also will start up the reactor after an emergency shutdown without significant reactor damage. In the case of significant damage (sever core accident), the reactor will not be started back up.

### **6.2.2 Shut Down**

The shut down transient can also be another high risk region where simulation has not been exercised. This will bring the plant to a shutdown state (e.g., when the fuel cycle is finished). It also will address emergency shutdown procedures.

### **6.2.3 Maneuvering**

These are the very gentle transients after start up and before shutdown when the reactor goes through minor adjustments to account for slight changes during the fuel cycle. This allows one to get the control rods in the correct position at the time of the initiating event.

### **6.2.4 Response to Initiating Event**

This is the emergency response to a failed component (i.e., what systems are shut off, what systems are turned on, and what the reactor operator is supposed to do in a given circumstance).

### **6.2.5 Human Factors**

This accounts for the human response of the nuclear plant operator. A human is not a robot and may vary from the operating procedures. These non-procedural responses can either increase or decrease the safety of the reactor. This brings the impact of human beings at the reactor into the safety analysis.

## **6.3 Aleatory Uncertainty**

This accounts for the statistical nature of component failure. Although we can continuously improve our ability to model when a component will fail from the capacity equations, there will still be the need to model some of the component failures from a purely stochastic point of view. The following list describes the key requirements of the aleatory uncertainty component.

### **6.3.1 Read Reliability Equations**

The reliability equations account for the probability of failure as a function of time (the bath tub model). These equations serve as a distribution to be sampled to account for the stochastic nature of component failure. This reliability information comes along with the plant component. At time zero (or the birth of a component due to failure and replacement or scheduled maintenance), the component sends the reliability information to the aleatory uncertainty component.

### **6.3.2 Role Dice to Choose Maximum “Lifetime”**

A random selection is made from the reliability equations to set the life expectancy from the reliability equations. This statistical model sets the aleatory life of the component. The aleatory life equations can extend this life if you are in a “good” environment or shorten your life if you are in a “bad” environment.

### 6.3.3 Set the Aleatory Life Clock

The aleatory uncertainty component then sets the end of life on the aleatory life clock. This clock ticks down the components life until it reaches the end. The component then fails.

## 7 Graphical User Interface

This is how the code user interacts with the software. To make it easier for twenty-first century nuclear engineers, it should look like the video games that they have been playing for years. The success or failure of production software depends mostly on the ease of use of the product. Accuracy of the spatial and temporal numerical methods or the elegance of the neutron transport model does not entice users to use a code. The key is how quickly and efficiently the code user can get his job done.

To improve the ease of use of the code, we will try to automate as much of the work required to perform a study. Most of the work of a convergence study or uncertainty analysis is very repetitive. This type of work is better done by a computer because it will be less error prone than a human being. Through automation, we will try to get as close to the “push a button and get results” mode as possible. One of the keys to improving input processing comes from doing verification, validation, and uncertainty quantification on each component. This way we will have a library of components that already have their pedigree established. When these pedigreed building blocks are combined, most of the verification of input is already done.

Another key step of reactor safety analysis is to compute a steady-state solution. The code will be designed to automate the steady-state solution and solve for it directly. Easy steady-state solutions are an advantage of starting from a fully implicit nonlinear solution method in the first place.

Finally, one of the key purposes of the GUI is to reduce the “user effect.” User effect is the difference in the solution caused by different choices made by two different code users who have been assigned to analyze the same transient. By automating as much as possible the input choices, the differences between two different users running the same transient can be made smaller. The following ten subsections (7.1 - 7.10) describe the important structures of the GUI.

### 7.1 Input and Output

The GUI needs to make setting up studies and analyzing the results as easy as possible. The goal will be to move as much of the menial tasks to the computer and away from the user. By utilizing the compute power available on cluster computing, large studies can be done relatively quickly. The data then needs to be processed and presented to the user in an easy to understand manner.

## 7.2 Tracking Decision Points and Sequence Spawning

This is the key component for the automatic creation of an event tree. In this mode, we will build in the ability to monitor all of the control model variables (both automatic and manual) that change the state of the reactor. Examples are opening a valve, turning a pump on, or SCRAMing the reactor. Whenever there is a significant change in a control model variable, the code will pause. The code will then spawn new processes to calculate the effects of the failure to change the control variable as expected.

The traditional PRA model is binary, meaning that the valve opens or the valve does not open. The code will then spawn two runs. In the first, run the valve opens; in the second run, the valve does not open. This will create a branch in the event tree. These two runs will proceed until the next decision point occurs. At this point, the code will pause and spawn more processes. Even if the code only spawns two processes at each decision point, the number of tree branches gets very large. This is why we need some form of a database to store and retrieve the required data.

There are two significant increases to the number of branches in the event tree: partial failures and failure dynamics. Partial failures increase the number of branches from a decision point from two to many. For example, a valve may open or it may fail closed. Additionally, the valve may fail at many different partially open positions. Each one of these partially open valve positions becomes a new branch in the event tree. Failure dynamics brings into account the timing of the event. The valve may open at 5 minutes, 10 minutes, or 20 minutes. This means that a single decision point can become many decision points at different times in the transient. The consideration of partial failures and failure dynamics explodes the number of branches in the event tree.

Another modification to the current fault tree will be the ability to have other than a binary end state of the event tree branches. Currently, there are two states, "OK" and "Core Damage." There are many transients of interest to the reactor owner that do not result in core damage but may result in a significant down-time for the reactor. The ability to estimate how much damage was done to the reactor, even if there was not core damage, will be computed as well. This information can help to address modifications in the operating procedures that have no impact on core damage frequency but have a significant impact on the profitability of the reactor. These simplified calculations will be done in the economics component.

The key to using this new capability will be to look at things in a graded approach. A fast calculation that addresses binary tree branches and binary end states should be done first. This study can be used to focus where more detailed information is warranted. The high run time cost modes of operation will need to be focused on areas where the pay-off is highest.

## 7.3 Study Initialization

One of the keys to ease of use is to automate, as much as possible, basic tasks that are often repeated. This process moves the load from the expensive human to the cheap computer. In addition, by freeing up more time for the reactor analyst, a better analysis of the generated data will occur. There are three main types of studies to be automated and they are discussed in the following sections.

### 7.3.1 Uncertainty Quantification

Uncertainty quantification can occur either through a black box Monte-Carlo approach or through the forward sensitivity analysis approach. Either way, this process will be automated so the user simply asks for a list of parameters to analyze and the results of the uncertainty quantification study are produced.

### 7.3.2 Optimization

Optimization can be done as either a Monte-Carlo approach or a Newton's method approach. Here, we are either optimizing the design to reduce risk or performing a vulnerability search. Again, the process will be automated so the user simply provides the function to be optimized and the resulting study is done and the optimized solution is presented.

### 7.3.3 Data Assimilation

Data assimilation can be done in two different modes. The first mode is based on static experimental data sets and the second mode is based on real time data acquisition. In the static mode the data are acquired through the database of existing experimental data. In the 4-D data assimilation (three dimensions and time), data are gathered from input streams with the code running in simulator mode. Again, this should be automated and require minimal user interaction.

## 7.4 Simulator Mode

Simulator mode is in contrast to batch mode. In batch mode, the code is given input and it simply processes its tasks until completion. In simulator mode, the code is required to respond to stimulus while it is running. This stimulus can come from the code user who will have the ability to interact with the model through the GUI. For example, the user can click on a pump in the reactor model and raise or lower the pump speed. In this interactive capability mode, the user becomes the "reactor operator" and he can see the plant response to different control model variables. If the code is fast enough to respond to these changes in "real time," then the code can be run in reactor simulator mode.

Additionally, there is another mode that requires the code to respond to stimulus during the run, this is called data assimilation. In this mode, the code is "wired" directly to the experiment. The code and the experiment run

simultaneously with the code accepting the experimental data “on the fly” and adjusting the static model parameters to force the code results to match the experimental data through the data assimilation process. This new method of computer code validation can be very useful. The experiment is run until the code is properly trained. This replaces the static experimental design process with a dynamic process. The real advantage of this approach is one trains the code until the code response matches the experiment’s response. When the code gets the “right” answer, you know that you have enough data.

## **7.5 Database**

There will be a large amount of data generated and required for a modern reactor safety simulation code. To improve the efficiency of the code user, this data will be stored in a database where it can be easily retrieved and used in the future. There are three main types of data that will be stored.

### **7.5.1 Inputs**

A significant amount of effort goes into building a description of a nuclear reactor. Because this input will describe traditional thermal hydraulics, neutronics, and PRA, this input description will be a significant investment. This investment will be protected by storing the information in a secure database where it can be retrieved easily by other users who need the information.

### **7.5.2 Outputs**

In the process of building the safety case for a nuclear reactor, there will be a very large number of code runs that provide supporting evidence. The ability to store this data in an easily retrievable form will allow one to be able to “drill down” into the details of the safety case when a higher level of detail is desired.

### **7.5.3 Validation Data**

Validation data supports the simulation code’s pedigree. The codes ability to match (and even better predict) reality is the largest contributor to the code users confidence in the software. The storage and protection of this data are important. We are now approaching a new form of validation where a high-resolution code calculation can serve as a numerical “experiment.” Due to the increasing cost and regulation of experiments and the decreasing cost of computers, this transition will accelerate in the future.

1. Experiments - it is important to store the raw experimental data, the processed experimental data, the as-built experiment drawing with the instruments clearly marked, uncertainty estimates of the data, and clear and accurate indications of the initial conditions and boundary conditions of the experiment, as well as any papers or write-up that describe the

experiment. This combination of data and meta data need to be part of the knowledge management process.

2. High Resolution Computation - one needs to store all code inputs, all code outputs, all processed code output, a write-up of the assumptions made in the model (e.g., isothermal or incompressible), the code executable (if not available, at least the code version), and what computer the simulation was run on. Again this needs to be stored in the knowledge management process.

## 7.6 Surrogate Construction

There will be parts of the analysis where high levels of uncertainty are acceptable. Examples are scoping studies where one is simply interested in trends and the goal is to decide where additional work should be done. For these types of studies, a fast running solution with low fidelity is the best approach. There are a variety of ways to construct simple fast running surrogates from higher fidelity calculations. This surrogate approach is often used in optimization and uncertainty quantification. Surrogate construction will be based on the following three-step process.

### 7.6.1 Low Fidelity

Because our application is different than other fields where surrogates are popular (e.g., the aerospace industry), we need to modify the basic approach. For aerospace applications, the solution space is relatively smooth and the equations are relatively linear. In reactor safety, the transient behaves differently before the pressure relief valve is opened than after the valve is open. The topology of the boundary conditions is different and the dominant physics is different. The opening of the pressure relief valve produces a sharp change in the problem solution. Linearly extrapolating the solution with the valve closed into the state space where the valve is open would produce very large uncertainties.

Therefore, the first step in construction of the surrogate will be to first use the low fidelity versions of each component to map out where significant changes occur in the solution. This is similar to automating the construction of the event tree. The low fidelity models connected to the control system in the code will define where the solution changes rapidly.

### 7.6.2 High Fidelity

Once the coarse grain map of state space is computed, state space can get subdivided into smooth regions. In these smooth regions, we will use experimental design strategies to choose how many high fidelity runs need to be done to create a surrogate with an acceptable level of uncertainty.

### 7.6.3 Unused Computer Time

If there are unused cycles on a computer where the code is installed, one can use the computer down time to improve the uncertainty of the surrogate. An experimental design algorithm can choose which high fidelity runs will result in the greatest reduction in uncertainty. These runs can be done in the “background” and the results can be stored in the database. These new runs can then be used to create a new surrogate that has a lower uncertainty but still runs fast.

## 7.7 Knowledge Management

This project will produce a large amount of information. Some of it will be experimental data, some of it will be output from code runs, and some of it will be automatically generated reports from different studies. In addition, there will be the software quality assurance documents, program plans, progress reports, conference talks, and peer reviewed journal papers. This information combined represents a large amount of knowledge that has been paid for by the taxpayers.

Historically, this type of information exists in separate places, with separate engineers, in differing levels of quality and retrievability. When one of these engineers retires or leaves the project, a large amount of “corporate knowledge” disappears. In this project, we will build into the project plan a knowledge management system. All information paid for on the project will be stored in a centralized system that can be read by anyone who has access privileges. There are three main advantages to this approach:

1. New employees - when a new team member is added, they are given access to the knowledge management system. They can quickly find the background information on the work and quickly come up to speed on the project.
2. Project shutdown - if the project has to be temporarily shut down, the information to restart the project is already stored in a safe place.
3. Moving the project - occasionally projects are moved from one location to another due to a switch in subcontractor - this knowledge management system will make these transitions efficient.

## 7.8 Economics Component

The purpose of this component is not to make detailed economic forecasts of the profit for the reactor owner operator: that work is well outside of the scope of this project. The purpose of this component is to put a simple constraint on optimization studies. For example, if one is optimizing operating procedures to minimize risk by maximizing the safety margin, there needs to be some form of constraint based on economics. Without an economic constraint on the solution, the safest reactor is one that is shut down and the components are continuously

replaced. To remove the absurd solutions from these studies, a simple economics model is proposed that consists of the following three pieces:

1. Cost - each component will include a replacement cost (parts and labor) and a replacement down time
2. Revenue - the amount of money made by the reactor will be an integration over time of the power level times a constant multiplier
3. Profit - this will be equal to the revenue minus the cost.

Although this model is very simple, it can be used to investigate changes in operating procedures. For example, “Can I keep the profit the same and reduce risk by replacing components at a higher frequency?” Similar to all of the other components in this software package, this model can be improved when funding, personnel, and a priority arises.

## 7.9 Check if the Simulation is Physically Reasonable

This is an important capability that serves as a sanity check on the code. Does the code obey the basic laws of physics and thermal dynamics? Is mass conserved? Is energy conserved? Is entropy increased? In steady state, does the sum of the pressure changes around the loop sum to zero? In steady state, is the amount of energy added to the closed system equal to the amount of energy removed? This capability is important for basic thermal dynamic analysis of the system and to ensure that the simulation code is working properly.

This capability also can be used to measure the quality of a surrogate. Because surrogates do not contain physics laws, there is no requirement that the surrogate conserve mass momentum and energy. How well the surrogate conserves mass, momentum, and energy is a measure of how well the surrogate has been trained. This physics check can be used to grade the quality of a surrogate-produced solution.

## 7.10 Experimental Design

This can be used in the future to support the design of experiments used for validation of the simulation code. In the short term, this capability will be used to train the surrogate. Here, one can make use of unused computer cycles to improve the quality of the surrogate. The experimental design algorithm can use down time on the computer to do runs of the code to improve the surrogates fidelity by reducing its uncertainty.

## 8 User

There is a shift in the type of code user that will be using this software versus the typical code user of the software written 30 years ago, like TRACE, TRAC, and RELAP. Historically, the code users of these legacy codes were capable of

reading the source code written in FORTRAN. These legacy code users also were very patient because they had to type their input onto computer cards and wait a few days before the code run finished. The historical code user also viewed the world though a very different paradigm. One had the choice of a very expensive computer that was slow or an expensive computer that was very slow. To this end, much of the analysis work shifted to the human to minimize the amount of work done by the computer.

Today, very fast computers are available with huge memories at a very low cost. It is far more cost effective today to shift the load from the human analyst to the computer. Many code users of today have not taken a class in any compiled computer language and are more comfortable with spreadsheets and high-level programming languages like Matlab. These users will not be able to dig into the source code to help look for problems. Additionally, these new code users are very comfortable interacting with a computer through a virtual reality system. This is how they learned computers through playing video games. These new users expect fast responses, easy to user interfaces, and much of the work to be automated; therefore it is done by the computer and not the human.

To entice these new code users, input processing will need to be similar to video game interfaces. The input should be fast, accurate, robust, and based on 3-D graphics. Results should come out directly as a report that can be cut and pasted into a document. Studies should be automated so the user provides minimal input on how the study is to be performed and the rest is automated, including summarizing the results.

There is another important note to make about the new code user. The current code users of the legacy codes have been running the same (or slightly modified) software on the same (or slightly modified) reactors for more than 20 years. These legacy code users have a large amount of high quality expert opinion based on years of work on the same reactors. For these users, the code simply verifies what their expert opinion predicts.

This project is creating new software with many new ways of solving problems. There will be a natural tendency of the older code users to stay with the legacy codes and the younger code users to drift to the new codes. Therefore, it is unlikely that the code users of this software will have significant expert opinion to guide them. Because of this lack of experience, the code processes need to be designed to minimize the importance of the code users judgment. This will naturally minimize the user effect. It is important to note that having well trained code users is the key to high quality analysis. One of the most important parts of this project will be focused on training code users to use the software efficiently and accurately.

## 9 Summary

This is a high-level requirements document that is not intended to provide detailed requirements (they will come later). It is designed to describe what work will be included in the software development portion of the project and

how those pieces will be fit together to produce a software package. Because of the newness of the approach employed in this software, there are still some details that need to be worked out. This short-term, focused research will be part of the work plan. Recognizing that although most of this work has been done in separate pieces at different times in different software, combining all of it together into a single piece of software will be a high-risk project.

To minimize the risk in development of the software, the project will be based on a test case approach. Software development will focus on a specific task (e.g., the main steam line break transient in a three loop Westinghouse PWR). For this specific task, all of the different capabilities will be demonstrated on a very focused problem. This focus will minimize the amount of software that needs to be developed because only a small part of the whole code is relevant for a given test problem. However, the test problem will clearly demonstrate how all of the design pieces fit together and one can see how the software works in a short timeframe (e.g., 2 years).

One of the keys to choosing the test problem will be to have a basecase that the new software can be compared against. In the simplest sense, this new software is a combination of two legacy software packages: RELAP5 and SAPHIRE. Therefore, the logical base to compare against is current RELAP5 and SAPHIRE results. There is a second advantage to choosing a test problem with existing RELAP5 and SAPHIRE inputs. The RELAP5 input is built in a slow, human-intensive process of reading blue prints and converting the information from the blue prints into RELAP5 input cards. However, the process of transferring RELAP5 input to a different simple reactor model is much easier. In addition, all of the event trees and reliability data are already available from the SAPHIRE input. All of the input data are in a single location, and we have RELAP5 and SAPHIRE experts available on hand to help translate the existing code inputs to a new input model.

To minimize the risk (or maximize the margin to failure) in the software development process, the software will be developed with check points built into the project plan. At each check point, all of the current software components will be combined together to make a system calculation. Based on these check-point calculations, new capabilities may be added and planned capabilities may be removed. Because this is a very new approach to reactor safety software, it is probable that the design pieces will not fit together. Here, we will discover that something was left out of the design. These omitted capabilities will have to be added to the requirements document in the future. Additionally, some of the planned requirements may be more difficult to implement than initially planned. These requirements will then need to be removed from the requirements document.

This high-level requirements document will be a living document throughout the software development. Requirements will be added and removed in a manner designed to minimize the risk to software development. However, once the list of generic capabilities, global physics, and components is solidified, detailed software quality assurance, verification, validation, and developmental assessment documents will be produced. These documents will then provide a

full pedigree of the quality of the software to provide the software user with confidence in the results.

It also is important to focus on the potential large pay-off value of this project. By combining capabilities that are scattered across many different software packages into a single tool, the cost saving to the code user will be significant. Additionally, a single description of the nuclear reactor that is used for all of the analyses (safety and risk) will prevent a large number of errors caused simply by inconsistency between the risk and safety input descriptions. By rigorously quantifying uncertainty, the quality of the solutions can be quantified and documented. By training a new group of nuclear reactor engineers to simultaneously consider risk, margin, and uncertainty in their decision making process, this project will have made a significant contribution to the safety of nuclear reactors in the future.

## **10 Acknowledgment**

Prepared for the U.S. Department of Energy Office of Nuclear Energy Under DOE Idaho Operations Office Contract DE-AC07-05ID14517 (INL/CON-07-12487). This work was funded by the Light Water Reactor Sustainability Research and Development Program, developed and sponsored by the Department of Energy. The report number is INL/EXT-10-18891.

## **A Table of Requirements**

This appendix contains a table of general requirements, their rational, and a reference to the section where they are discussed in the document. The order is not prioritized, it is simply the order that they appear in the document.

Table 1: Requirements 1 of 2

#	Requirement	Justification	Section
1	Components will be based on a generic component.	This provides standardization for components.	2.1
2	The load equations will have multiple fidelity.	This improves run time and UQ.	2.1.1
3	External events will be included in the load equations.	This allows external event risk to be analyzed.	2.1.1
4	The load equations will be nondimensionalized.	This helps accuracy and UQ.	2.1.1
5	Capacity equations will be added.	This models physical component failures.	2.1.2
6	The capacity equations will be nondimensionalized.	This helps accuracy and UQ.	2.1.2
7	Reliability equations will be included.	This models stochastic component failure.	2.1.3
8	Alatery life equations will be included.	Minor adjustments to statistical failure due to physics.	2.1.4
9	An aleatory clock will be included.	This determines one mode of component failure.	2.1.5
10	All parameters will be input with a range and distribution.	This is required for UQ, DA, and optimization.	2.1.6
11	A list of global parameters will be maintained.	This will insure accurate global parameter UQ.	2.1.6
12	Control model variables will be included.	This models the control system and the operator.	2.1.7
13	Input processing will be done in 3-D.	This enables multiple fidelity levels that are consistent.	2.1.8
14	Initialization will be done as a restart.	This separates the solver from the GUI.	2.1.10
15	Three modes of initialization will be included.	This provides flexibility for analysis.	2.1.10
16	The component numerics will contain three dimensions.	This allows for dimensional collapse.	2.1.11
17	One cell in a direction turns off that dimension.	This is how dimensional collapse will be done with input.	2.1.11
18	The equations will have the ability to turn off physics terms.	This is how fidelity changes will be included.	2.1.11
19	Interfaces shall be built off the generic interface.	This provides standardization for future interfaces.	2.2
20	The code will replace failed components.	This enables long-time transients.	2.2.1
21	The operating and maintenance procedures will be codified.	This model accounts for humans at the reactor.	2.2.2
22	Human factors will be included in the operating procedures.	This accounts for human variability in responses.	2.2.2
23	Uncertainty quantification will be built in from the beginning.	This allows for more detailed studies.	2.2.3
24	Uncertainty quantification will resolve space and time.	This allows UQ focused model development.	2.2.3
25	Data assimilation will be a built-in study.	This allows for easier data assimilation.	2.2.4
26	Optimization will be a built-in study.	This allows for more detailed optimization studies.	2.2.5
27	An automatic control system of set points will be included.	This models the reactor control system.	2.2.6

Table 2: Requirements 2 of 2

#	Requirement	Justification	Section
28	FOM evaluation will be built in.	The FOM drives many different studies.	2.2.7
29	The code shall be structured based on Fig. 2	This provides a segregated development process.	2.3
30	Real time speed is very desirable.	This will enable a reactor simulator mode.	2.3.1
31	The software will include a database.	This will enable manipulation of large amounts of data.	2.3.2
32	Validation data will be stored with the code.	This makes code validation easier.	2.3.3
33	A simple economics component will be included.	This provides constraints for optimization studies.	2.3.4
34	General fidelity interfaces will be included.	This allow many code developers to contribute.	2.3.5
35	Composite components will be enabled.	This allows for complex components.	2.3.5
36	Global physics and local physics will be separated.	Physics local to a component can be added easily.	2.3.6
37	Global physics subject matter experts are required.	This provides experts for component construction.	2.3.6
38	Validation experiments will be designed to reduce uncertainty.	The most benefit for slow and costly experiments.	2.4.1
39	SQA procedures will be followed for production software.	This structures the software development process.	2.4.2
40	The software should be open source.	This allows for university involvement in development.	2.4.3
41	All components exchanging fluid will have 1one EOS.	This prevents inconsistent fluid properties.	4.1
42	Material properties will include aging effects.	The enables life-extension capabilities.	4.2
43	Fluid equations will have multi-fidelity.	This enables speed and UQ.	4.3
44	Neutronics will have multi-fidelity.	This enables speed and UQ.	4.4
45	Cross section generation will be user friendly.	Ease of use.	4.4
46	Cross sections will include burn-up.	This enables long-time transients.	4.4
47	Components will be as specific as possible.	The enables efficient SQA.	5
48	Control system will have fuzzy logic.	This minimizes numerical errors impact on the solution	6.1
49	Time step selection will be influenced by the control system.	This enables implicit control for long transients.	6.1
50	Decision points will be tracked and new process can be spawned.	This enables the automatic creation of an event tree.	7.2
51	Partial failures of components will be enabled.	This enables analysis of partial failures impact on risk.	7.2
52	Dynamic timing of failures will be enabled.	This enables analysis of the failure timing's impact on risk.	7.2
53	Non-binary end-states of the event tree will be enabled.	This will enable more detailed analysis of the reactor risk.	7.2
54	Surrogates will be constructed.	This enables run time speed when needed.	7.6
55	The GUI will check the physics of the solution.	This enables grading of surrogates.	7.9