Chapter 12
Bayesian Inference for Multilevel Fault Tree Models

12.1 Introduction

Fault tree modeling is an approach used to organize (in a graphical fashion) the various failure causes of a particular system. Typically, fault trees start with an undesired “top event,” then combine the various ways the top event can occur using AND and OR operators (i.e., logic gates). A fault tree model is a static system representation that analyzes system failure behavior from a deductive logic point of view. That is, given a failure at a specific level in the fault tree, the question is asked: how could this failure have occurred?

A fault tree for a system is composed of “basic events” that are inputs to the logical gates. And, it is at this level of basic events where data and other information are commonly used, via Bayesian inference, to estimate component failure probabilities. In other words, the vast majority of fault tree analysis performed in support of PRA over the last 40 years has employed Bayesian inference at the *lowest level* in the fault tree model. However, information on system and component performance may be available at higher system “levels”. For example, we might have failure information at the system, subsystem, or component levels. Also, we might have failure information for a group of components, where any one piece-part in this group can fail the group—this grouping is typically called a “super-component.” We demonstrate how information at any level of a fault tree model can be combined via Bayesian inference to estimate failure probabilities at any level of that same fault tree model.

12.2 Example of a Super-Component with Two Piece-Parts

In this first example, assume we have a super-component that is placed in an operational demand situation, where we do not have any prior information on this component’s behavior in such a setting. For illustration, we will assume the

Table 12.1 OpenBUGS script for representing a super-component with only top-level failures represented

```
Single super-component, Jeffreys prior on p.TE
Demanded 10 times, No failures
model {
# Assign model for system top (TE) observable (x.TE number of failures)
x.TE ~ dbin(p.TE, n.TE)
# Assume Jeffreys prior for TE failure probability
p.TE ~ dbeta(0.5, 0.5)
}
data
list(x.TE=0, n.TE=10)
```

Jeffreys prior on the component demand failure probability. We then collect operational data, where we demanded this super-component ten times and saw no failures.

As discussed in Chap. 3, the Jeffreys prior for a binomial aleatory failure model is a beta (0.5, 0.5) distribution, and the prior mean is $0.5/(0.5 + 0.5) = 0.5$. After performing the Bayesian update (using OpenBUGS) with this conjugate prior distribution, we find that the posterior failure probability mean value is 0.045. The script for this calculation is shown in Table 12.1.

This failure probability mean value (0.045) represents the "top level" of information (AKA, the super-component, or the "system"). Based upon the binomial aleatory model, and the associated data, we estimate the super-component performance is 0.045 (in this example, we focus on the mean value, but in reality we would have use of the entire posterior distribution for the failure probability).

Now, consider that we really had two piece-parts inside this single super-component. We would now like to estimate their respective failure probabilities. For the case of modeling failures at a lower level (where operational data is available), additional engineering information may be used to supplement the operational data. For example, assume a super-component is known to comprise multiple piece-parts. Further, we only have operational data for the super-component (e.g., we do not know which piece-part caused observed failures). It is possible to model the piece-parts *without* having direct observed operational data for these parts—and additional engineering information may be incorporated (part 1 is more reliable than part 2, part 3 is less reliable that part 1, etc.) into the modeling approach. Bayesian inference (with modern tools) is able to handle multiple sets of information simultaneously, regardless of the "level" of the modeling [1].

Returning to our two piece-part super-component, we can model the piece-parts as being a single system, where either part may fail the system (or super-component). The results of this simple system model (using OpenBUGS) would yield a posterior mean value (on a piece-part) of 0.024, where the applicable script is shown in Table 12.2.

Table 12.2 OpenBUGS script for a super-component which is made up of two piece-parts

```
model {
# Assume Jeffreys prior for subcomponent failure probability
p ~ dbeta(0.5, 0.5)
# Assign model for super-component
x.TE ~ dbin(p.TE, n.TE)
# Construct the overall fault tree structure (series system)
p.TE <- 1 - ((1-p)*(1- p))
}
data
list(x.TE=0, n.TE=10)
```

If we knew that each component was demanded on each of the ten super-component demands, then we would pool the data, so we have data consisting of no failures in 20 demands. This pooled data yields (using the Jeffreys prior) a mean failure probability (piece-part) of 0.024, the same as in the inference case above. This pooling of data for parts (at the lowest level of a fault tree) is the typical approach to PRA. However, we would need to perform the Bayesian calculation using the approach as described in the OpenBUGS script for this example if:

- We could not pool the data.
- We did not know the number of actual demands on the piece-parts.
- We wanted to incorporate other engineering information (e.g., part A is more reliable than part B).

12.3 Examples of a Super-Component with Multiple Piece-Parts

We could carry out the inference process for any number of piece-parts, for example if we had three piece-parts, we would see that the mean value (of a piece-part) was 0.016. Consequently, we can subdivide a super-component into *n* piece-parts in probability space and keep the same overall level of information at the top of the fault tree, that is, at the super-component level. It is important to note that in these OpenBUGS calculations, the entire distribution is estimated, not just the mean value.

In the example above, we encoded observational information at the super-component level, where in our example we did not see any failures at this level. In that situation, the information (no failures) is "shared" equally between the piece-parts since we did not include any additional engineering information (e.g., Part A is more reliable than Part B). If we see failures of specific piece-parts or if we have additional engineering information, these could be factored into the Bayesian inference process.

In the next example, assume we have a super-component with three piece-parts. However, we experienced a single failure in the ten demands. When a

Table 12.3 OpenBUGS script for a super-component with three piece-parts, where one has failed

```
3 piece-parts, Jeffreys prior on p_demand on each
Demanded 10 times, 1 failures on Component A
model {
# Assume Jeffreys prior for subcomponent failure probability
p ~ dbeta(0.5, 0.5)
p.B <- p # SOK dependence
p.C <- p # SOK dependence
x.B ~ dbin(p.B, n.B)
x.C ~ dbin(p.C, n.C)
p.A ~ dbeta(0.5, 0.5)
x.A ~ dbin(p.A, n.A)
# Assign model for supercomponent
x.TE ~ dbin(p.TE, n.TE)
# Construct the overall fault tree structure
p.TE <- 1 - (1-p.A)*(1- p.B)*(1-p.C)
}
data
list(x.TE=1, n.TE=10, x.A=1, n.A=10, x.B=0, n.B=10, x.C=0, n.C=10)
```

super-component fails once in ten demands, its posterior mean failure probability is 0.14 (again, assuming we started with the Jeffreys prior).

Now, if we move to a lower level in the fault tree for the super-component and evaluate the three-piece-part case when Part A fails, we see the following posterior results (note that the prior at each level of the tree was assumed to be the Jeffreys prior):

- The failure probability mean value (piece-part A) = 0.11.
- The failure probability mean value (piece-part B, C) = 0.019.

The OpenBUGS script for this example is shown in Table 12.3.

Again, note that in this example, the overall super-component level information is preserved (its mean failure probability is 0.14) but Part A is much more unreliable than the other two parts, which is consonant with the information that has been collected.

In the case where we have additional engineering information, say at the piece-part level, this information can be factored into the Bayesian inference calculation. Assume we have the super-component (with three piece-parts) as in the last example. Again, we see a single failure (Part A) in ten demands of the super-component. However, for this example, assume that we had information that indicated (prior to collecting any data) that Part A was five times *more reliable* than B or C. For example, assume that we believed the prior failure probability for Part A was given by a beta(1, 9) distribution, which has a mean value of 0.1 (or five times lower than the mean of the prior for Part B and Part C, which are still using the Jeffreys prior).

Table 12.4 OpenBUGS script for a super-component with three piece-parts (one has failed), but where one part is believed to be more reliable than the others

```
model {
# Assume Jeffreys prior for the piece-part failure probability
p ~ dbeta(0.5, 0.5)
p.B <- p
p.C <- p # SOK correlation
x.B ~ dbin(p.B, n.B)
x.C ~ dbin(p.C, n.C)
p.A ~ dbeta(1, 9) # MORE reliable A component case (prior)
x.A ~ dbin(p.A, n.A)
# Assign model for supercomponent
x.TE ~ dbin(p.TE, n.TE)
# Construct the overall fault tree structure
p.TE <- 1 - (1-p.A)*(1- p.B)*(1-p.C)
}
data
list(x.TE=1, n.TE=10, x.A=1, n.A=10, x.B=0, n.B=10, x.C=0, n.C=10)
```

Using this informed prior for Part A, the Jeffreys prior for Parts B and C, and the one failure of Part A out of ten demands, we see the following posterior results:

- The mean value (piece-part A) = 0.091.
- The mean value (piece-part B, C) = 0.019.

The OpenBUGS script for this example is shown in Table 12.4.

Note that the posterior information for Part B and Part C has remained unchanged, but the mean failure probability for Part A is slightly lower, owing to its lower prior probability.

12.4 The "Bayesian Anomaly"

In the case where we are modeling various levels of a system, it is important to note that simplistic or naïve analysis can lead to misleading results. For example, one specific situation that has been referred to as the "Bayesian anomaly" focuses on the improper application of the multilevel modeling approach [2, 3].

To explore this issue, assume we have the super-component with three equal piece-parts. However, assume that we evaluated each of the three piece-parts independently from one another. Thus, an individual piece-part has a posterior failure probability mean (after ten demands, no failures) of 0.045.

Now, we wish to evaluate the next higher level, that of the super-component. So, we "OR" the three piece-parts (i.e., the super-component is failed if Part A fails or Part B fails or Part C fails). The super-component posterior mean then is:

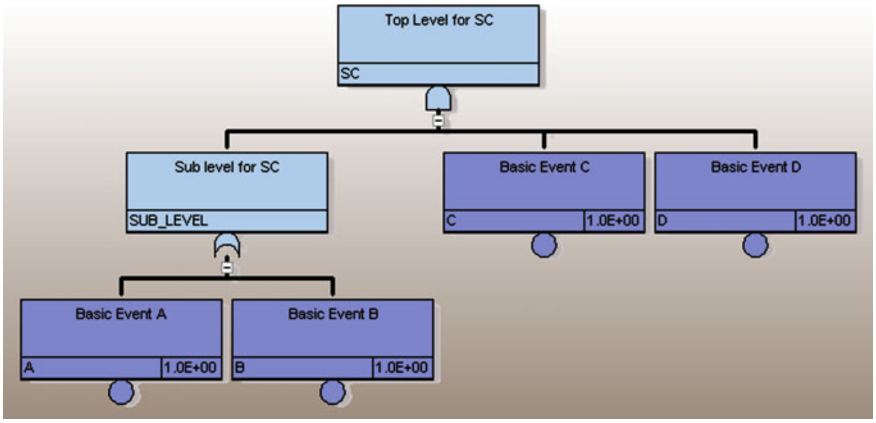

Fig. 12.1 Example super-component fault tree consisting of four sub-components

$$p(\text{super-component fails} \mid \text{no observed failures}) = 1 - (1 - 0.045)^3 = 0.13.$$

However, we know this to be incorrect (from our earlier calculation) since the super-component failure probability should be 0.045 (from Sect. 12.2). In effect, by treating each component as *knowledge-independent* of the other components, we are including additional information (e.g., from the Jeffreys priors for the three parts) that is not valid since the three parts share a state-of-knowledge correlation that is ignored via the simplistic analysis. As the number of piece-parts is increased, the simplistic analysis becomes even less accurate. Consequently, we need to be careful how high-level information is partitioned down to a lower level. However, moving from low-level information up to a higher level is not as problematic.

12.5 A Super-Component with Piece-Parts and a Sub-System

While we have demonstrated the ability to “flow down” data and information from a high-level (e.g., a super-component) to lower levels (e.g., piece-parts), the ability to represent multiple different levels in a complex system is a useful feature of Bayesian inference. To further illustrate this concept, let us explore a slightly more complicated example by assuming we have the super-component that is represented by the fault tree in Fig. 12.1. For this fault tree, the observed operational data are at multiple “levels,” including the sub-system and the super-component level rather than just at the component level.

Further, assume we have data for the gate named “SUB_LEVEL” (the sub-system) and the top event labeled “SC.” Specifically, assume there have been:

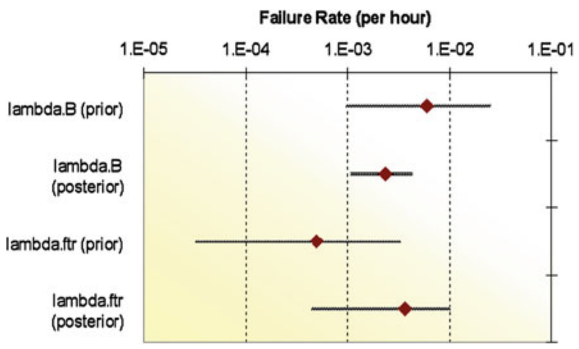

Fig. 12.2 Posterior distribution summaries for components B, C, and D (mean and 90% interval)

- Three failures of gate SUB_LEVEL in 20 demands (of sub-system level, not the entire super-component).
- One failure of the super-component in 13 demands of the system.

Note that the sub-system level (SUB_LEVEL) was also challenged during the 13 super-component demands (since the only way to have a super-component failure includes a component failure from this sub-subsystem). For the component level, assume:

- Basic events "A" and "B" represent a standby component, which must change state upon a demand.
- Assume that "A" represents failure to start and "B" represents failure to run for the required time (100 h).
- Basic events "C" and "D" represent normally operating components (100 h run time).

We will assume, for this calculation, the following prior lognormal distributions for the basic event parameters:

- Basic Event A: Mean 0.001 and Error Factor 5.
- Basic Event B: Median 0.005/h and Error Factor 5.
- Basic Event C: Median 0.0005/h and Error Factor 10.
- Basic Event D: Median 0.0005/h and Error Factor 10.

Note that since we use the same prior distribution to represent epistemic uncertainty for events C and D, we will have to account for this state-of-knowledge dependence in the Bayesian inference. The OpenBUGS script to perform this calculation is shown in Table 12.5.

Bayesian inference for this example yields a marginal posterior distribution for each of the four basic events. These posterior distribution sumaries show that the observed operational data has significantly increased the mean failure rate (lambda.ftr) for components C and D. The posterior mean of the failure rate (lambda.B) of component B has decreased from its prior mean value, while the mean failure probability for component A saw only a small change. An illustration of the components that did change is shown in Fig. 12.2.

Table 12.5 OpenBUGS script for the fault tree shown in Fig. 12.1

```
model {
# This is system (fault tree top) observable (x.TE number of failures)
x.TE ~ dbin(p.TE, n.TE)
x.Gate.E ~ dbin(p.Gate.E, n.Gate.E)
p.TE <- p.Gate.E *p.C*p.D     # Probability of Top Event (from fault tree)
p.Gate.E <- p.A + p.B - p.A*p.B     # Probability of Gate E from fault tree
p.C <- p.ftr   # Account for state-of-knowledge dependence between C & D
p.D <- p.ftr   # by setting both to the same event
p.B <- 1 - exp(-lambda.B*time.miss)
p.ftr <- 1 - exp(-lambda.ftr*time.miss)
# Priors on basic event parameters
p.A ~ dlnorm(mu.A, tau.A)
mu.A <- log(mean.A) - pow(log(EF.A)/1.645, 2)/2
tau.A <- pow(log(EF.A)/1.645, -2)
lambda.B ~ dlnorm(mu.B, tau.B)
mu.B <- log(median.B)
tau.B <- pow(log(EF.B)/1.645, -2)
lambda.ftr ~ dlnorm(mu.ftr, tau.ftr)
mu.ftr <- log(mean.ftr) - pow(log(EF.ftr)/1.645, 2)/2
tau.ftr <- pow(log(EF.ftr)/1.645, -2)
}
data
list(x.TE=1, n.TE=13, x.Gate.E=3, n.Gate.E=20, time.miss=100)
list(mean.A=0.001, EF.A=5, median.B=0.005, EF.B=5, mean.ftr=0.0005, EF.ftr=10)
```

As illustrated in the previous example, we are able to mix both failure mechanisms (e.g., fails to start, fails to run) in addition to different levels of information in our Bayesian inference approach. For more details on this "multi-level" approach to Bayesian modeling see [4, 5].

12.6 Emergency Diesel Generator Example

To further demonstrate the general Bayesian approach, we will address another example related to modeling of a single emergency diesel generator (EDG). In the simplest case, one might represent the failure probability of an overall EDG as a constant for each test (even though the EDG is composed of hundreds of piece-parts). However, this assumption is not really accurate since different tests for the EDG may test different parts of the EDG in different ways. For example, some tests may only require a short run time, while others require a long run time. Further, the long run time will require additional demands on some components [e.g., fuel transfer pump (TP)] that may not be used during short tests. A simplified fault tree for the EDG in this example is shown in Fig. 12.3.

We have included three EDG failure modes in this fault tree:

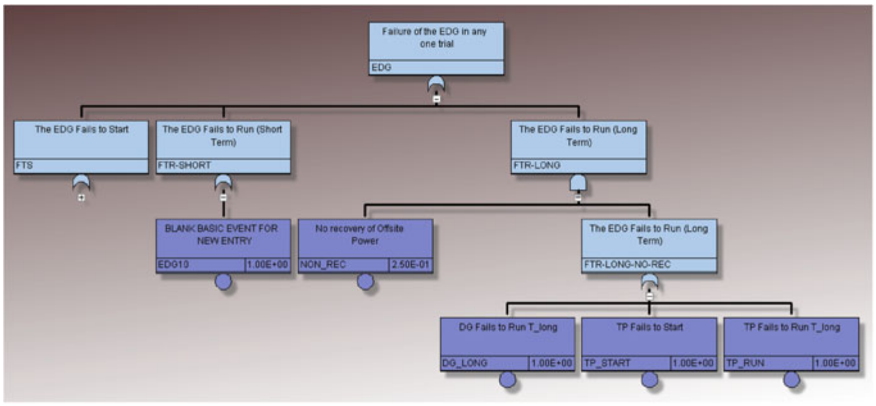

Fig. 12.3 Fault tree structure representing the EDG super-component and constituent components

1. Fails to start (FTS).
2. Fails to load/run (over a short duration) (FTR.S).
3. Fails to run (long duration) (FTR.L).

Assume that the time periods of interest (for the running failures are:

- t_short = 1 h.
- t_long = 24−1 = 23 h.

Assume that the EDG fuel oil transfer pump (TP) is demanded (exactly) six times during long tests. Note we could relax this assumption and model the number of demands probabilistically, as discussed in Chap. 10. Also, for simplicity we do not include offsite power recovery or repair from a failed state.

The uncertain parameter in the aleatory model for each basic event has an assumed prior distribution:

- lambda.short ~ gamma(0.495,164 h) (the CNI prior for failure to run).
- lambda.long ~ gamma(0.5, 625 h) (the CNI prior for failure to run).
- lambda.TP ~ gamma(0.5, 100,000 h) (the CNI prior for failure to run).
- p.TP ~ beta(0.498, 498) (the CNI prior for failure to start).

Then, in order to obtain the posterior distributions, we require operational data. Assume we have seen:

- 33 short tests (1 h duration) with no failures.
- 3 long tests (24 h) with 1 failure—TP failed to start.

Running this case with OpenBUGS (with the single TP failure) gives (posterior mean values):

- Lambda.TP = 5.0E-6/h (TP fails to run).
- Lambda.short = 2.5E-3/h (DG fails to run short term).

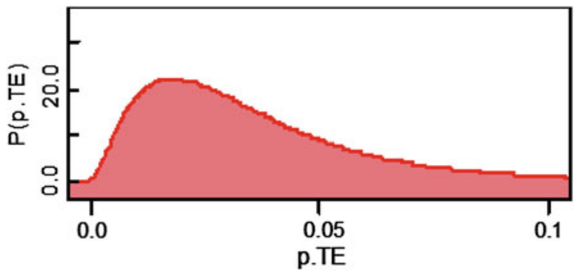

Fig. 12.4 Posterior density function for the EDG fault tree top event

- Lambda.long = 7.5E-4/h (DG fails to run long term).
- p.TP = 2.9E-3/demand.
- EDG = 3.6E-2/test (p.TE: top event EDG fails), the marginal posterior density function for p.TE is shown in Fig. 12.4.

The top contributors to the failure probability of the EDG top event are (in order)

- p.TP_START = 1.7E-2.
- p.DG_LONG = 1.7E-2.
- p.DG_SHORT = 2.5E-3.
- p.TP_RUN = 1.1E-4.

12.7 Meeting Reliability Goals at Multiple Levels in a Fault Tree

We could extend any of these examples to determine to what degree we are meeting a reliability goal, set at any level of the fault tree. In order to demonstrate this concept, we will revisit the example from Sect. 12.3. In that example, we had a super-component with three piece-parts and we observed a single failure in ten demands. When the super-component fails once in ten demands, its failure probability posterior mean value is 0.14 (again, assuming we started with the Jeffreys prior). For the lower level in the fault tree, we found:

- The failure probability posterior mean value (piece-part A) = 0.11.
- The failure probability posterior mean value (piece-part B, C) = 0.019.

Assume that we had the following reliability goals:

- The system reliability is greater than 0.9.
- Each piece-part reliability is greater than 0.99.

The OpenBUGS script for this example is shown in Table 12.6:

The posterior results (monitoring the `p.meet` nodes) indicate:

- The probability of meeting our system reliability goal is about 33%.

Table 12.6 OpenBUGS script to determine whether reliability goals are met at different levels in a fault tree model

```
model {
# Assume Jeffreys prior for subcomponent failure probability
p ~ dbeta(0.5, 0.5)
p.B < - p # SOK dependence
p.C < - p # SOK dependence
x.B ~ dbin(p.B, n.B)
x.C ~ dbin(p.C, n.C)
p.A ~ dbeta(0.5, 0.5)
x.A ~ dbin(p.A, n.A)
# Assign model for supercomponent
x.TE ~ dbin(p.TE, n.TE)
# Construct the overall fault tree structure
p.TE <- 1 - (1-p.A)*(1- p.B)*(1-p.C)
# Compute goal nodes
TE.unrel.goal <- 1 - TE.goal
PP.unrel.goal <- 1 - PP.goal
p.meet.TE <- step(TE.unrel.goal - p.TE)
p.meet.PP.A <- step(PP.unrel.goal - p.A)
p.meet.PP.B <- step(PP.unrel.goal - p.B)
p.meet.PP.C <- step(PP.unrel.goal - p.C)
}
data
list(x.TE=1, n.TE=10, x.A=1, n.A=10, x.B=0, n.B=10, x.C=0, n.C=10)
list(TE.goal=0.9, PP.goal=0.99)
```

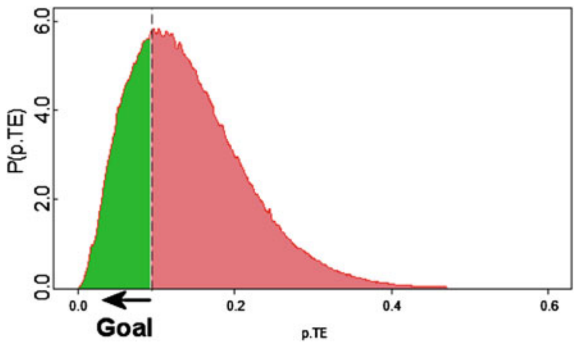

Fig. 12.5 Posterior density function for the fault tree top event showing the chance of meeting a reliability goal (shown in *green*)

- The probability of meeting our piece-part reliability goal for A is about 2%.
- The probability of meeting our piece-part reliability goal for B is about 50%.
- The probability of meeting our piece-part reliability goal for C is about 50%.

The posterior distribution for the top event is shown graphically in Fig. 12.5, where the goal is placed at an unreliability of 0.1 (i.e., a reliability goal of 0.9

implies an unreliability of 0.1). The green area of the curve is the probability of meeting the reliability goal, which in this example is about 33%.

In this section, we described how information and data may be available at various levels in a fault tree model, and how these may be used within a Bayesian analysis framework to perform probabilistic inference on the model. Initially, we describe this approach using a simple fault tree model containing a single top event and two sub-events. The simple model represented a super-component and two piece-parts. For this system, we showed how modern Bayesian analysis tools (specifically the OpenBUGS software) may be used in a relatively straightforward manner to fully describe the probabilistic information known about the system. Then, we extend this approach to more complicated systems and sets of information.

Lastly, we showed how this same analysis tool (OpenBUGS) can also be used for the example models to determine the probability of meeting a reliability goal for any level in the fault tree model. In general, reliability goals are desirable system performance levels for driving improvements—not meeting specified goals is generally a flag indicating degraded performance.

References

1. Wilson A, Huzurbazar A (2006) Bayesian networks for multilevel system reliability. Reliab Eng Syst Saf 92:1413–1420
2. Guarro S, Yau M (2009) On the nature and practical handling of the Bayesian aggregation anomaly. Reliab Eng Syst Saf 94:1050–1056
3. Philipson L (2008) The 'Bayesian Anomaly' and its practical mitigation. IEEE Trans Reliab 57(1):171–173 March
4. Johnson V, Mossman A, Cotter P (2005) A hierachical model for estimating the early reliability of complex systems. IEEE Trans Reliab 54(2):224–231 June
5. Reese C, JohnsonV, Hamada M, Wilson A (2005) A hierarchical model for the reliability of an Anti-Aircraft missile system. UT MD Anderson cancer center department of biostatistics working paper series, paper 9, October