Modelling Uncertainty for Requirements: the Case of Surprises

Dylan J. Walton, Huma Samin, Nelly Bencomo
Department of Computer Science
Durham University, UK
{*dylan.walton, huma.samin, nelly.bencomo*}*@durham.ac.uk*

***Abstract*—The concept of Surprise has been used to model uncertainty for self-adaptive systems (SAS). The uncertainty in the environment of SAS demands the system to perform self-adaptation decisions, and therefore makes it hard to formulate, validate and manage their non-functional requirements (NFRs). A number of probabilistic measures exist that compute surprise to flag up situations of uncertainty for NFRs. A problem with these measures is that they don't give any information about how big or small the surprise is, and therefore lack support for quantification of the level of uncertainty and its impact on NFRs. The challenge here is to classify the size of surprise to better model uncertainty levels for NFRs. We argue based on classification, surprise can be used to identify failure situations for NFRs, and thereby support modelling of risks. In this paper, we propose a framework to allow for the classification of surprise. Based on this framework, we perform an analysis into risk for the NFRs. As a proof of concept, we have applied the framework to case of Remote Mirroring. The risk analysis is performed on both individual NFRs as well as combinations of them to demonstrate the framework's utility across a variety of tasks.**

***Index Terms*—Uncertainty Modelling, Surprise, Self-Adaptive Systems, Risk Analysis, Non-Functional Requirements**

I. Introduction

Surprise has been used in many fields, in neuroscience for example surprise refers to a series of reactions to startling or otherwise unexpected events [1], [2]. This wide range of applications has lead to many different definitions and formulations of surprise tailored to given tasks where surprise might be used. Surprise in computation, like in neuroscience, can be defined as a difference in expectations where the expectations are approximated using different variables in a system [3]. These expectations, also known as beliefs, could potentially vary wildly, the system needs to know how to react appropriately to large changes in expectation which are "surprising". Surprise naturally arises from uncertainty, and has been used as a method to model uncertainty for self-adaptive systems (SAS) [4], [5].

The uncertainty in the environment of SAS demands the system to perform runtime self-adaptation decisions, and thereby manage its non-functional requirements (NFRs) [6]–[10]. Hence, a SAS should be able to autonomously assess deviations from its specified behaviour and use these deviations to consider triggering the corresponding adaptations. Surprise has been used here to provide a measure of this discrepancy and can then be used to help designers deal with this uncertainty by providing measures of how new information differs from a designers and a systems expectations [5]. Hence, the concept of surprise allows the SAS to be able to deal with uncertainty in an explicit way.

A number of probability based measures have been developed for computation of the surprise [1], [3]. Out of the measures presented in [3], Bayesian Surprise [2] has been used to flag up situations of uncertainty for NFRs [5], [11] of SAS. A problem with the existing measures is that they don't give any information about how big or small the surprise is, and therefore lack support for quantification of the level of the uncertainty and its impact on the satisfaction of NFRs. This leads to a potential problem with surprise, which is how can we classify the size of surprise? We argue that having the capability to measure the size of surprise can help the SAS to identify and model the situations of risks that could affect the satisfaction of the NFRs, and therefore improve the adaptive mechanism.

In this paper, we propose that surprise can be used to produce a model of risk for SAS. We propose a set of techniques to classify surprise to model uncertainty for the NFRs of SAS. Based on the classification of surprise, we perform risk assessment using the level of uncertainty identified by the surprise classes. For the purpose of experimentation, we have used two types of surprise measures: Bayesian surprise and Confidence Corrected surprise (CCS). So far Bayesian surprise has been the dominant way to calculate surprise to measure uncertainty [4], [5], [11], but it is not the only measure and other types of surprise [1], [3] can capture different facets of information that could help both in modelling risk and its analysis. CCS differs from Bayesian surprise as CCS attempts to bring in confidence to the surprise value [3], which is done by using a single distribution as its "prior"; Bayesian surprise in contrast uses multiple priors at different points of the process (priors will be explained in section 2). Both measures of surprise will be used and contrasted with their helpfulness in modelling risk within the SAS.

The contributions of this paper are as follows:

1) First is the framework which will be used in how to analyze and interpret the different surprise values, specifically with how they relate to risk.
2) Related to the framework is the creation of techniques that can be used to link specific groups of surprise to increased risk, the results of which are what will be interpreted by the framework.

3) This paper will also look into the classification of different surprise values based on their potential risk within the environment and the potential risk they may indicate to the SAS itself.
4) The last key contribution of this paper will be an evaluation into the use of alternate surprise measure i.e. Confidence Corrected Surprise (CCS) [1].

The paper will show the results of the tests into these contribution on an illustrative case within an example SAS, the SAS being the Remote Data Mirror Simulator (RDMSim) [12]. The RDMSim will contain all relevant metrics required to calculate surprise and demonstrate the results. Experimentation consists of applying the techniques in order to identify "high risk" groups, for a technique to be successful it means that different groups have unique features that differentiate them from other groups that is not simply due to size alone. Experimentation on classification will involve tweaking parameters in order for the end result to resemble the differences in the groups, so what ever is classified as high risk has features that would indicate this risk, the reverse being true for low risk groups. Experimentation on CCS vs Bayesian surprise as discussed will be performed by applying the identified techniques to both and finding which is more helpful for risk modelling and analysis. It will be shown that there are a number of techniques that can be used to find groups at different levels of risk, that these groups can be classified according to some criteria, and that CCS can be used as an alternative to Bayesian surprise, though its application may vary in context. The results of these sections will all contribute to the framework produced that will allow requirement engineers to better analyse risk within their systems.

The paper is organized as follows: Section II covers the baseline concepts, fleshing out important concepts related to surprise and uncertainty in SAS. Section III covers the methodology, where the techniques to be employed are explained, as well as how classifications are determined, how both the results link to risk modelling. Section IV covers the experimentation, where tests with the different possible techniques are covered to check their effectiveness, as well as tests into the specific ways classification are performed. Section V covers discussion on the results found in experimentation, as well as potential applications elsewhere. Section VI covers related work to this topic, and section VII concludes the paper.

II. Uncertainty in SAS and the concept of Surprise

The environments that SAS operate in are by their very nature subject to uncertainty [6], [7], [9], which is to say that these environments are unpredictable and our systems may be subject to events that designers could not foresee [6], [7], [13]. That is not to say it is impossible to make predictions, whilst environments might be random they are most likely still constrained by certain rules, or follow trends that designers can take advantage of, uncertainty should mostly be an issue with edge cases that happen somewhat infrequently, or due to many small errors building up over time. A SAS when reasoning about its environment maintains a belief (or assumption) about the environment. This belief is based on several variables maintained by the SAS that give some indication about the current state of the surrounding environment and what the SAS thinks will happen next in the environment. These beliefs, for example, could be about the satisfaction of NFRs such as reliability or performance of a component of the system, or about some estimated variable within the environment, but all are given as a probability that this event does/does not occur. It should be briefly noted that beliefs are not equivalent to expectations, though for the purposes of this paper they will be treated as such as the concern is only with those beliefs that are also expectation.

For the purpose of modelling and reasoning about uncertainty of SAS, the concept of Surprise has been presented [5], [11]. In animals, surprise arises from the difference between prior expectations and the reality as the events occurs [2]. Similarly, for the SAS, a surprise is defined as the difference between prior and posterior beliefs about the state of the environment [5]. The question becomes what is to be used as prior and posterior beliefs, which is answered through the use of Bayesian surprise [4], [5] defined as follows. Bayesian surprise is one way of mathematically formulating surprise. It is measured as the distance between prior and posterior beliefs [3] [5], with prior beliefs simply taken to be beliefs before an event occurs, and the posterior beliefs those immediately following the event as follows:

$$S_{BS} = log_2(\frac{P(m|D)}{P(m)}) \tag{1}$$

Where m is simply some belief and $m|D$ is the same belief after being modified by some data (D) or series of events.

This can be contrasted to something like confidence corrected surprise (CCS) [1] which is another probability based surprise measure that we have used for modelling of uncertainty for SAS. CCS is defined with a prior belief as some flat, uniform, distribution meant to provide information relating to confidence on account of the single prior; and posteriors as before are taken to be the beliefs at any subsequent timestep after some event/series of events has occurred [3]. The reason for using a flat distribution is because surprise should be larger when a system is confident about a given belief, which is captured by the different between the flat prior and the posterior beliefs. This flat belief will be taken to be the belief at the very beginning of a run, which will always be a 50% chance of failure. Confidence corrected surprise offers an interesting comparison to Bayesian surprise, the latter might be sensitive to many rapid changes due to the prior constantly changing with the posterior, the former on the other hand by using a single prior can potentially offer better insights as all surprise values will relate to a single point. That is not to say that CCS is guaranteed to produce more reliable surprise values, rather that it and Bayesian surprises use might circumstantial based on the issues a designer is facing at that time. The CCS is computed as follows:

$$S_{CC} = log_2(\frac{P(m|m_{flat})}{P(m_{flat})}) \quad (2)$$

Where $m|m_{flat}$ refers to any given belief (the "flat" belief after being transformed by new data) and m_{flat} refers to the flat distribution here taken to be the belief at the first timestep (50% chance of failure).

III. Methodology

This section presents the methodology for the proposed approach. We have used an example case of Remote Data Mirroring (RDM) System [14], [15] to illustrate the full methodology. Next, we describe the RDM case as follows:

Example Case: Remote Data Mirroring: A remote data mirroring is a disaster recovery technique which involves maintaining links to backup server from which lost information can be restored or otherwise accessed [14], [15], its successful operation will require it have different requirements balanced. As a concrete example, we have used an open source artefact of RDMSim [12] to provide simulations of the different environmental situations for a remote mirroring network. The RDMSim is designed to assist designers of SAS by providing an environment for which they can use to test their managing components (these being the parts of the SAS that make decisions regarding the environment). The RDMSim itself require 3 non-functional requirements (NFR) be satisfied which relate to the performance of difference parts of the system, the Minimization of Operational Cost (MC) requires bandwidth consumption not reach a threshold; Maximization of Performance (MP) requires that the time taken to edit the contents of the backup servers (i.e. both reading and writing time) not reach a threshold; lastly is the Maximization of Reliability (MR) which require a minimum number of active links to be maintained. The architecture of RDMSim comprises two main components: the managing component which involves the adaptation logic and the managed component which involves the application logic for the remote mirroring network. As the managing system involves the adaptation logic for the RDMSim network, it maintains beliefs (i.e. assumptions) relating to the different requirements, with the beliefs giving the probability a given requirement will be satisfied. Finally the RDMSim also allows for tests to be done on different scenarios which can simulate different potential environments a SAS may find itself in.

The process of modelling uncertainty for NFRs based on surprise, and the risk analysis for performing analysis of failure situations of NFRs, presented in Fig. 1, is described as follows:

A. Surprise and Failure

An important part of this analysis will be the use of surprise to assess the situations of failure where some requirement is unsatisfied. In the RDMSim, failure will be where at least one of the three NFR's crosses their threshold and ceases to meet its required satisfaction level. The way surprise and failure are linked is that a surprising event (but not surprise!) will be determined to be where failure occurs, more specifically where a previously satisfied NFR switches to being unsatisfied. The reason for using where failure first occurs over failure more broadly is it can give better insight into the risks that lead to failure, rather than also having values that simply continue with the trend established by previous events. This will allow a clear link between surprises and risks as now we can produce subsets of surprise values that are also guaranteed to be surprising and use these to look for issues both in the environment but also in the managing system of the RDMSim simulator. From this point on failure will be used to refer to the surprising events rather than simply any time step some requirement is not met. Potential subjectivity is dealt with using the 3 NFR, the idea is that each NFR gives a subjective measure of the systems satisfaction, with is possible all, none, or some combination being satisfied.

The first part of the process to model uncertainty for NFRs is the calculation of surprise using the equations shown in section 2. The beliefs are taken from the RDMSim [12] simulator and as previously discussed represent the probability of failure for one of the 3 NFR's as assigned by the simulator and managing system. The beliefs maintained by RDMSim at each simulation time step are iteratively used to calculate the Bayesian [5] and Confidence Corrected Surprise [1], producing a list as long as the initial input. As the Bayesian Surprise and Confidence Corrected Surprise represent a deviation value without an upper bound. They don't give information of how big or small a deviation or a surprising situation is. For the purpose of computing the size of the surprise, we have used a Sigmoid function [16] which bounds these surprise values between 0 and 1. For specifically CCS, the values are also normalized by subtracting the smallest element, this is to avoid instances where all values are incredibly high and thus are all moved into the same grouping. Also produced by the simulator is a list of simulation time steps where failure occurs, these values are also loaded and stored in a list such that each NFR surprise value and has an associated "failure" which is either true or false, this list is then edited such that only values where failure occurs and the previous state was satisfied are considered to get surprising events. From here a subset of surprises can be produced which gives the surprise values for where failure occurs, thus the original set and its subset can be used to start checking which surprise values have an affinity for failure, the methods for refining these groupings is discussed in the next section. This array of surprising events can be used to produce a subset of surprises where surprising things happen, risk can then be modelled based on comparisons between the two.

B. Surprise identification

One of the contribution of this paper is the identification of techniques to model high risk groupings from the list of surprises, the first of these techniques is a simple binning method to start classification of the surprise values. Here, the surprise values are grouped in some intervals according to the size of surprise, and as surprise value is between 0 and 1 so are the interval bins. This technique allows us to further

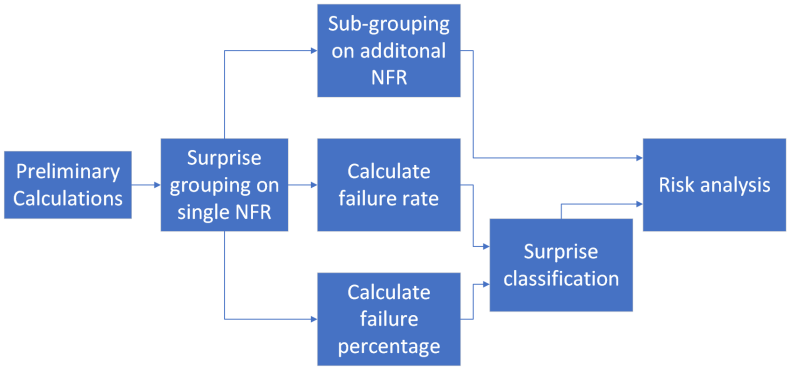

Fig. 1. A broad outline of how Uncertainty is modelled for Risk Analysis.

focus on the separate bins allowing a more thorough analysis of the contents. We can also ”bin” the results of the subset of surprises where failure occurs in the same way, and using these two values we can calculate the failure rate of a given group (that is, the percentage of elements within the group that are surprising) and the percentage of failure for a group (what percentage of failure in the entire run belongs to a given group). To illustrate the difference consider some group that has 20 elements out of 100, and 10 failures of a particular NFR out of 40, in this instance the failure rate would be 50% as half of the values in the group are failures, the failure percentage however would be 25% as the group contains that amount of all failures in the entire run.

The next method for identifying high risk groups for classification is by breaking down what is happening in the other NFR when failure occurs. What is done here is that for one of the groups produced in the binning, we can break down this group further using the exact same technique. However, now we will be looking at the surprise values of the other NFR to see if there are any links that might help us identify risk. This technique is usable for all types of surprise, and it is hoped that it will allow us to identify high risk groupings and can also provide knowledge about how different NFR impact each other.

C. Surprise Classification and Risk Assessment

The next stage in the process is the classification of surprise for the purpose of risk assessment. This involves taking the groupings produced by the prior steps and classifying them based on risk, which here will be taken to be the number of failures over the total number of surprises in that group interval. From here the classifications can be refined and changed, the alternate methods such as using subsets related to increases in risk can also be considered and grouped, and other factors may be considered to add weighting to the classifications, such as having too few instances to make a proper judgement or having a disproportionate amount of surprising events (even if these surprising event to surprise group ratio is low) may be considered to add alter the classification in some ways.

As risk has been modelled to be based around the likelihood of failure classification can now be performed using the results of the techniques previously mentioned, the end result should be a classification that factors in both risk within the SAS's operational environment and risk to the SAS itself through poor belief calculation. Risk in the environment is given simply by the failure rate and failure percentage previously discussed, it involves no further analysis on why the order of risk is the way it is, it simply takes it at face value and used this to decide which surprise values are likely to contain failure. Risk analysis can be performed by simply checking the classification as well as any other saved metrics taken from the techniques and using this to assess where resources should be allocated. Analysis into risk due to errors within the SAS is done using the same classification but adding an additional step, checking how far away different groupings are from where they ”should” be, some discrepancy is expected (such as with groups that simply never occur because the average surprise is too low/high) but a lot of difference or even inversion of the expected order could point to errors. Analysis here can be used to identify areas for improvement and issues within the SAS when engineering requirements.

D. Bayesian and Confidence corrected surprise

Previously, Bayesian surprise has been the dominant form for which surprise has taken in modelling uncertainty for SAS [5], [11]. This is not the only form of calculating surprise, so this paper extends its scope to also compare Bayesian Surprise to a different technique, namely Confidence Corrected Surprise (CCS). A comparison between CCS and Bayesian surprise will also be produced by comparing how well the surprise values can identify failure, which versions require the least work to find these surprise groupings, and which produce the most ”intuitive” surprise classifications at the end of the process. These can be measured based on which definitions produce the groups of surprise with the highest proportion of failure, which produces the most high surprising groupings, how many steps it takes to find a high failure surprise group, and finally by comparing the produced classification to the intuitive one as described in the baseline concepts section. The comparison intends to find which of the two definitions is most helpful for modelling risk, which may be both generally or scenario specific.

IV. Experiments

In this section the proposed techniques will be tested for each scenario of the 7 scenarios of the RDMSim [12]. Due to the limitation of space, only one (scenario 0) will be discussed in this paper, however the remainder can be found in a GitHub repository [17]. In order to ensure results are accurate 5 runs over 500 time steps each will be performed, the relevant metrics will be calculated and averages of the 5 runs will be shown. The techniques tests will be done for both Bayesian and Confidence Corrected Surprise, with a brief comparison done for the illustrative scenario case. The experimentation for the proposed techniques consists of applying the techniques as proposed in the methodology and looking for unique features within groups. What is meant here is if the techniques were to fail there would be very little to no difference between the set of values produced after applying a technique than the set of values it is being applied to, success means the technique has produced unique values that differ from the base set and provide insight into which groups are likely to cause/contain failure. Experimentation for classification consisted of tweaking a process for moving groups into classes, this set of experimentation is very subjective and so the details of tweaking do not go into detail, the important part is that classes reflect the results produced in techniques, that something that has high chance of failure is correctly move into one of the two highest groups. Experimentation into Bayesian vs Confidence Corrected surprise is done parallel to experimentation into techniques, techniques applied to both sets of surprises will be shown and briefly compared, though the actual discussion is saved for the subsequent section.

As mentioned in the methodology surprise values will be binned, what was chosen were 10 equal bins from values 0 to 1 in increments of 0.1. The lower bound of some interval will be used as shorthand for the entire range, so the group 0.2 to 0.3 will simply be referred to as 0.2.

A. RDM Illustrative Scenario Case

Basic binning was performed on the now calculated surprise values as discussed in the methodology and as shown in Fig. 2, the first set of results will focus on the helpfulness of failure rates. Failure rate was found by dividing the number of surprising elements in a group by the number of all elements in the group, and the results are promising. For MC in Bayesian surprise two groups had a higher than 25% failure rate, 0.6 and 0.7, with 0.6 having the highest at 73% and 0.7 having 34%; MP had groups 0.6 and 0.8 with failure rates of 62% and 42% respectively. The results for CCS were similar, for MC a single group was identified that had more than 25% failures, that being 0.5 with 73% failures, MP had 0.6 and 0.9 both of which also had 62% and 42% failures. The results on these values are very similar, CCS faltering only in producing the additional high failure group Bayesian does, with the difference being made up that the remaining groups in CCS all have slightly higher failure rates than their Bayesian counter parts. In both cases the failure rates only occurred for the smaller groups however, with larger ones having rather low failure rates which limits the use of the technique here. Unremarked upon so far is MR as the results differ from the rest, for Bayesian surprise groups 0.3 and 0.4 had failure rates of 51% and 41%, with the remaining results trailing off as surprise gets lower. CCS separates all surprising values into 3 groups, each of which had around 50% failure. In both cases for MR the groups are large, high failure groups accounting for around 2/5 values, CCS seemingly worked slightly better as it perfectly separated failure so the only groups to have it for MR all had above 45% and having more members in these groups. This also starts a trend in which MR differs in behaviour from the other two NFR, which is partially expected as the other two NFR are linked, but the extent to which this differs stands in contrast to how the manager should behave, this now serving as an early indicator that something is wrong. Importantly, this value can be used to help model risk in the environment as it gives the probability that if this surprise value is found that it will mean failure but cannot be used alone to issues with the SAS.

Another simple technique is to simply see which groups contain the highest percentage of total failure, with groups containing a large amount of failures still posing a risk even if the failure do not make up a large percentage of the group itself. This technique on its own can be used to help understand risk within the environment, like before by indicating which groups are likely to contain failure, but this method can be combined with a comparison to the size of percentage of members each group has relative to the total number of potential members in order to gain better insight. An example of this comparison is shown in Fig. 4, where for groups in MP 0.6 has a much larger amount of failures compared to the groups normal size. Results here were again successful, for Bayesian surprise MC 0.4 contained half of all surprise values which should indicate its potential risk, however, 0.4 itself makes up half of all elements in a run normally so the extent of this risk is mitigated by this fact. In CCS MC 0.3 was effectively equivalent to Bayesian MC 0.4, both Bayesian and CCS had groups that had failure occur at a much higher rate than the group itself did, those being 0.6 for Bayesian and 0.5 for CCS. The results between CCS and Bayesian are again similar, as are the results of this technique and the prior one, and though it will be shown in other scenarios that this is not guaranteed to be the case it does point to many similarities between the methods of computing surprise, that even though they use completely different priors a broad trend of types of groups emerge between them when conditions are right.

Techniques thus far have focused on identifying risk within the environment by finding which groups of surprises are likely to have or even be failures, a different use is to find faults within the managing component itself. Methods here are looking for differences between which surprise groups are actually likely to indicate failure and which ones we expect to indicate failure, where there is a difference there is a potential error within the SAS. The expected ordering here is a simple size order, as larger surprise if functioning well means a larger probability of failure, and though this ideal situation is unlikely to be followed precisely larger gaps can point to

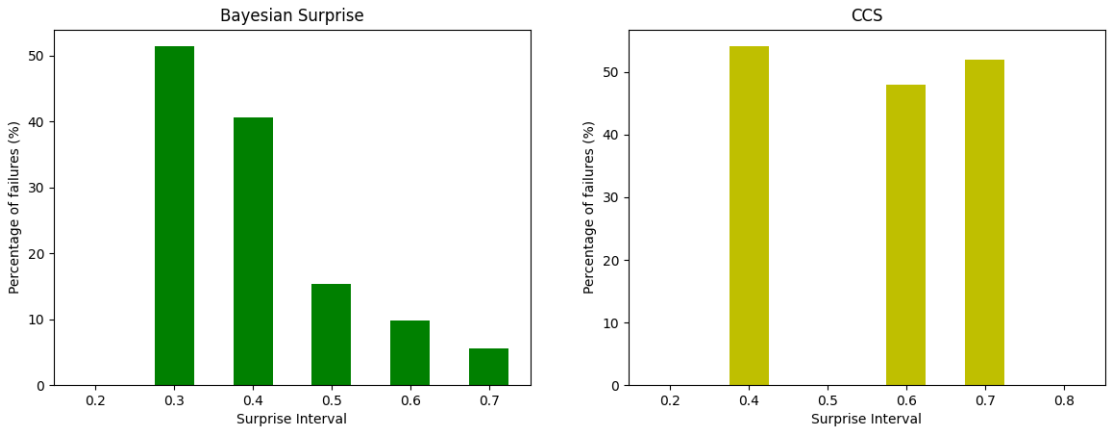

Fig. 2. The different failure rates for the groups of surprise in Bayesian and Confidence corrected surprise

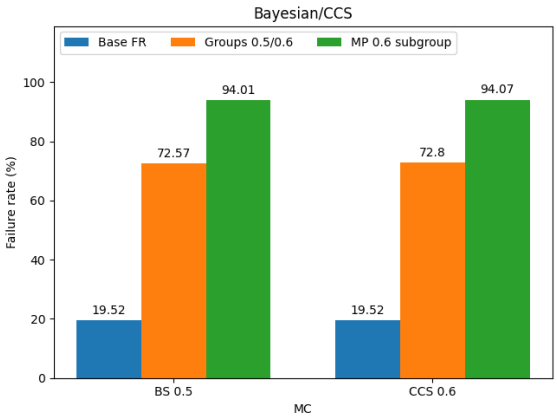

Fig. 3. A comparison between different potential groupings, using the base rate of failure for all surprises in MC, Bayesian group MC 0.5 and CCS group MC 0.6, as well as the subgroup formed from MP 0.6 for both/

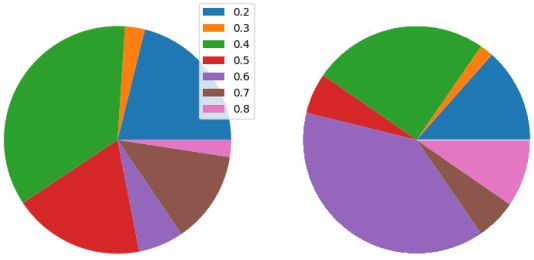

Fig. 4. The different proportions of each group with more than 5 values for MP. On the left: The size of each surprise group as a percentage of the total. On the right: The size of each failure subgroup as a percentage of the total. Legend shows which groups are used for both.

serious issues. Within scenario 0 the most obvious example is MR, as previously discussed MR values in Bayesian are inverse to the intuitive size order, with risk given by failure rate decreasing as surprise increases, which would mean that events that beliefs say decrease the probability of failure are in reality more likely to fail than later events. Comparing this to the other NFR which behave relatively more like the expected version this means something is going wrong when making decisions for MR, possibly that it is being under-prioritized. CCS had a similar problem though not the extent Bayesian surprise had it, in CCS as discussed the three groups to contain surprise all had high failures rates around 50%, with at least one group detailing a slight decrease in risk (group 0.4), this is not as severe however as 0.4 was small compared to the other high risk groups. In this instance CCS again served to produce better indicators of risk, though the technique is useful for both types of surprise as evidenced by Fig. 3.

The last technique to be discussed is the breaking down of groups into further subgroups based on the surprise values of other NFR. This technique can be used for classification but its main purpose here is to assist with analysis and link together the different NFR than purely serve as additional categories. The results were incredibly successful, for many different groups across the different NFR's and surprises subgroups were identified that suggested strong links between different surprise values, whether they had failures or not. MC 0.2 for example in CCS had well over half of its values (171) occur alongside MP 0.3, for Bayesian surprise a similar thing occurred between MP 0.4 and MC 0.4, where 149 out of 176 values for the former were shared with the latter. In terms of producing sub groups more indicative of failure success was found but for this scenario was slightly limited, where high surprise subgroups were possible it was usually only as improvements to groups that were already classified as having a high failure rate, where such an improvement was possible it was often significant. For Bayesian MC 0.6 had a failure rate of 73% for 35 values, the subset made by also using values that were members of MP 0.6 had 24 values but a failure rate of 94%. Interestingly the reverse was not true, MP 0.6 MC 0.6 had a similar failure rate to base MP 0.6, though the potential of the method was still demonstrated, and both values still occurred most frequently alongside each other. This was also true for CCS MC 0.5 and MP 0.6 behaved similarly to MC 0.6 MP 0.6 in Bayesian surprise, here however there

TABLE I
RISK CLASSIFICATION

Risk Level	Surprise Grouping
1	0.3, 0.5
2	0.7
3	0.4
4	–
5	0.6

was also a pairing between MC 0.5 and MR 0.5, where this subset also had a failure rate of 94%. The subgroups were not included in classification on account of their small size, most were below 20 members and though they were significant as these results were confirmed over multiple runs may clutter the classifications at the moment for little gain. That is not to say that they can never be used for classification, but simply that in these runs it would inappropriate to do so.

B. Classifications

A lot has gone into developing and analysing these techniques described so far, but the next stage in the process is the classification of results. The first step involves looking at two metrics, the internal failure rate of a group, and the difference between that groups occurrence, and the occurrence of surprises within that group as members of their respective sets (occurrences generally and the subset of failure surprises). An initial sorting is done based purely on the failure rate, which depending on the percentage of failures can be sorted into one of five groups, corresponding from low to high risk. The bounds of these groups are in intervals of 12.5% until 50%, which due to the few examples of groups beyond this values everything beyond 50% is grouped as high. Sorting is not yet finished however as this is not the only metric. Also relevant to consider are the total percentage of failures a group has as well as the number of members, based on which values may be moved up or down in risk. An example is some group may be sorted into the "middle" risk grouping due to having a failure rate above 25% but below 37.5%, the proportion of surprise however may be quite large, and in this instance it would be moved up a grouping as well. The final step is considering the actual number of values, if the group is too small (which is here is defined to be less than 10 instances) it is deemed too small to be of great risk and simply moved to the lowest value, if the amount of members is less than 20 the group is simply "demoted" to the risk group below as whilst it may pose a great danger its values are still too small to consider it equivalent to groups with many members. The groups should thus give levels of risk relating to both risk in the environment. The classifications of risk for both Bayesian and Confidence Corrected Surprise on the NFR MC, with numbers 1 to 5 corresponding with low to high risk (Values that are low risk due to lack of values are left out) as shown in Table I.

These values on top of telling us about risk within the operating environment can also tell us about faults in the SAS, though they are few in this example. The key issues are the low value for 0.7 and the fact that 0.4 and 0.5 should in likelihood be reversed. In the former case this is a result of its size, though due to having enough values to be relevant should still be just below 0.6, this may be simply because of noise and a stricter filter is required, or this may be because of faults within the system, though designers would now know this is an area to look into. The fact that 0.5 is below 0.4 despite both being large may point to the system over prioritising MC, hence why values immediately increasing risk slightly are so low, it could also be a case of under-prioritisation where risk decreases but not by much.

C. Framework and interpreting results

Mentioned during the demonstrations of the different techniques is how exactly these are then linked to surprise and risk analysis, some early connections should be apparent, such as the focus on identifying groups with high failure which would have the obvious advantage of highlighting to designer where attention needs to be paid. The main technique to accomplish is then is simply calculating the failure rates of the different groups, which would give an indication of how much certain surprise values are to indicating a failure in an NFR. Another connection to risk analysis is the use to find potential faults within the managing component of the RDMSim, as discussed the RDM Simulator is supposed to assist designers of SAS in designing managing components that can balance multiple and potentially conflicting NFR. A key part of this will be the calculation of beliefs, which if designed appropriately will mean that if a belief in failure increases from one time step to another a failure is actually more likely to occur. Using a different technique involving the comparison of the percentage of items a group contains compared to the rest, to the same but for the subset of values where failure occurs, the prior technique of simply using failure rate for the group can also be used. What is expected is that the larger surprise is, the more failure should be in that group, either as the groups failure rate or simply by containing a larger proportion of failure than any other grouping. Discrepancies between this expectation and the actual results can point to errors in the managing component, and the way these errors manifest in the different NFR can point to what the precise cause is (if has low failure throughout but another has many failures where the should not perhaps it will be the case that manager is biased towards one NFR in a way it should not be).

V. DISCUSSION

A. Efficacy of the Approach

The binning of the surprise values allowed for an easy and intuitive way to break down the surprising situations in order to analyze the subsections. The bins then served as the basis for which further analysis could be performed focusing on failures and their size relative to the different groups. A limitation of this technique is that there is no real way to decide an appropriate size of a bin nor how it should be broken up. It could be the case that smaller and more frequent bins can work better by breaking up larger groups, it is shown in the results for the other scenarios that certain bins contain too large a

share of values which caused issues when trying to analyse the results. Smaller intervals, however were not a guaranteed fix, using 20 bins for example did not help much for certain scenarios when breaking up groups, and further sub-divisions risked breaking up any other successful groups until it was too small to be relevant. A different approach to binning may have also helped, the use of machine learning models was considered and tested in order to more accurately bin data using techniques such K-means clustering [18], though these the groups produced were often less intuitive and were not as successful at finding groups that should discrepancy with how the manager should be running. Nevertheless, it may still be the case that machine learning methods [19] can employed for the early groupings, though here it proved to be ineffectual.

Next is the use of said bins for identifying potential risks in the system, which was done by comparing the number of failures in group to the number of elements generally within that group, with slightly different ways of performing this depending on if the environment or the system itself was the focus of the analysis. For analysis into the environment the main methods were comparison of failure as a percentage of the interval group (so comparing what percentage of the group was failure) and comparisons between the size of the group as a percentage of the total (number of group elements over number of elements generally) and the number of failures as a percentage of its total. As discussed in the risk assessment part of the methodology, these two factors can give insight how likely a given surprise value is to indicate some failure has occurred, and where failures are most frequently occurring, which can allow designers to better identify risk and determine how risky certain surprise values are (though this links to the classification). There is also an ”intuitive” way of analyzing surprise, which is the larger surprise is on at a given timestep the more likely failure is to occur. The technique comparing failure rates and broader failure occurrence only has use due to the fact the expectations of how surprise values are to behave are not always met. This can be for benign reasons, perhaps certain surprise values never occur and so seemingly appear low risk, or they are outliers but can be safely ignored, sometimes however these differences can indicate a more fundamental problem with how beliefs are assigned and in this case the second use of the techniques can be used. Like with the environment focused method the comparison between group percentage and the corresponding group percentage for the subset of values where there are failures may be used, this and simply comparing the order of ”least likely to most likely to have failure” serve as the analysis into the managing system itself. The goal was to find how these high failure rates differ from the intuitive understanding, and based on this further classify ”risk” based on likely faults in the managing system. This would assist designer by help pointing to potential flaws in their current designs. The main drawback here is that whilst it points to the presence of risk, and potentially even gesture at the cause based on which surprise values ”misbehave”, it does not give a clear indication of what precisely causes the risk.

To assist in the previous two ways of identifying risk was an additional technique focused on combining multiple NFR surprise values, rather than having its own purpose it could help analysis and classification by linking together different NFR by producing subgroups that may be more/less prone to failure, as well identify which values occurred alongside the different NFRs. It has been demonstrated in the illustrative case alone the potential use, with many different surprise groupings being dominated by one or sometimes two groupings in a different NFR. The new subgroups can assist in identification of failure in the environment by producing specific groups more prone to failure, and can also potentially assist in identifying risk in the managing system by pointing to potentially related reasons in the other NFR. An example case for the latter might be that unexpected failures are occurring in a low surprise value for the NFR MR, but that this occurs only when MC also has a low surprise value (but no failure) in this instance a potential problem may be the over-prioritisation of MC

B. Efficacy of Surprise classifications

The other key proposal of this paper is the classifications of surprise, as discussed in experimentation this is done in multi-step process, the first step involves producing initial assignments based on the failure rate of certain groups and percentage of general failures. These are separated into 5 classes which assign risk low to high based on both risk in the environment and risk for the the system, after the initial assignment several other factors were considered to specific details of the classification. If the number of group members for example was too low this would risk demoting moving the group down a class or simply assigning it to low risk if the group was so infrequent, an additional movement up the classes could also be made if the proportion of surprise was large enough. The hope for the classifications is that designers can quickly use the methods to find parts of the managing system that need most urgent attention when dealing with risk. A difficulty with this method is how to precisely weight certain sections, the current method is rather simple as when considering number of occurrences two simple bounds consider, if it the group has less than 20 occurrences it is demoted, if it has less than 10 it is moved to the lowest risk group. Alternative methods may include even a variable weighting system, where fractional movements can be done based on number of members of a group, rounded to some whole number, though for the purposes here were there were 500 members and usually only 2 or 3 groups had more than 100 members this seemed redundant.

C. Bayesian Surprise vs Confidence Corrected Surprise

The last thing to be discussed is the use of Bayesian surprise vs CCS for the purpose of risk analysis, first is the use for identifying identifying risk within the environment. Focusing on identifying groups with high failure rate the results were mixed, as both Bayesian and Confidence corrected could produce such groups. Early on however, confidence corrected appeared to be slightly better producing more ”unique

groups",which is to say groups that differed from the base surprise rate, which meant some groups had relatively low surprise but others had really high surprise. This remained true for Bayesian surprise as well, but was often slightly less so than CCS. For MR values in particular, CCS in particular could produce groups that were both large but had concerningly high failure rate, such MR 0.6 in the illustrative example given in experimentation. This trend however flips for later scenarios, for both the different failure rates become more uniform, but this is particularly true for CCS, Bayesian on occasion at least produced some variation that might better point specific areas of concern. A similar tend was noted when simply looking at the proportion of failure each group had relative to the total, early on CCS would produce groups that contained over half of all failures, whereas (depending on the scenario) Bayesian surprise produced groups which spread out failure, the key downside being that it made it more difficult to identify the causes of failure by having groups less indicative of failure. Sometimes this was simply due to CCS producing larger groups, such as was the case for all NFR in scenario 2, in these instances not much new can be learned which would not already be know from Bayesian, and in fact in these scenarios Bayesian me even be better due to the fragmented failures allowing for more variation, where focus can be given on more specific values within the groups than simply having to look at one half of the runs.

The other part was using the above techniques to identify risk within the managing component itself, for both types of surprise this was focused on identifying discrepancies between how groups should be ordered and how they actually were, and if there were other factors that might mitigate this. The results again were also mixed, in that which surprise measure was better varied from scenario to scenario. Both types of surprise could produce groups and orders that could potentially indicate flaws in the system, in scenario 0 the order of groups with the most surprise for MR was in fact reverse to what it should be on average over the 5 runs, CCS for the same scenario however had all failure concentrated in a single groups, whose proportional size was twice that of the regular groups including non-failures as well.

VI. Threats to validity

Two different types of threats to validity will be covered, those being external and internal validity. Briefly, external validity concerns threats to performance and the scalability of the solution; internal validity covers whether the proposed solution will perform well in an actual environment.

External Validity: Currently we have test the system using 3 NFR, however situations may exist where we have more NFR which may pose a threat to external validity. This could potentially pose a threat to performance if the number of NFRs was to greatly increase, though it is our intention in the future to expand on this and test for greater numbers of NFRs.

Internal Validity: For internal validity, the results achieved thus far have been ran on a simulated environment rather than an actual one to simplify the process. Further tests will be done to ensure the results of the RDMSim reflect an actual environment. In a similar vein, findings in this paper should also be confirmed for different types of environment, though the RDMSim is supposed to be quite generic in order to assists a variety of designers knowing these techniques interact with other specific configuration could help highlight pitfalls or improvements of and on these techniques.

VII. Related Work

A number of techniques have been presented in the literature to deal with uncertainty for SAS [8], [20], [21]. For the purpose of quantification of uncertainty, Bayesian Theory of Surprise has been used to deal with uncertainty related to NFRs for SAS [5], [11]. Bayesian surprise has been used along with the approaches of DDNs [4], [5] and POMDPs [22]. Moreover, in [11], Bayesian surprise has been used with multi-attribute analysis and Pareto Analysis to provide support for multi-criteria decision making (MCDM). The approaches presented in [5], [11] makes use of Bayesian Surprise as a flag to identify the uncertain environmental situations. They don't perform analysis based on the size of the surprise to model the level of uncertainty. In contrast, our proposed framework makes use of both Bayesian Surprise [2] and Confidence Corrected Surprise [1] to measure the level of uncertainty by identifying how big or small a surprise is. This uncertainty level helps us in performing risk analysis.

Moreover, to specify uncertainty in the requirements specifications, a formal requirements language known as RELAX has been presented [23], [24]. Taking the semantics of RELAX as a base, a technique known as RELAXing Claims has been proposed [25]. The RELAXing Claims approach studies the effects of uncertainty, taking into account the problems in the monitoring infrastructure, on the validity of Claims (i.e. assumptions/beliefs of the environment of the SAS). Furthermore, based on the concept of RELAXing Claims, a technique called REAssuRE [7] has been developed. The approach of REAssuRE makes use of goal models and Claims to support decision-making in SASs. For the purpose of studying positive and negative impacts of uncertainty on the runtime configurations of SAS, an approach called POssIbilistic SElfaDaptation (POISED) has also been developed [9]. The POISED approach is based on possibility theory and fuzzy mathematics. The POISED approach is used to deal with uncertainty for improving the quality requirements (NFRs in our case) of SAS at runtime. Moreover, techniques based on probabilistic model checkers have also been used to specify and analyse properties to perform reasoning about uncertainty [26]. The technique presented in [27], [28] facilitates the software architects to specify uncertainty about the effect of prospective alternative adaptation strategies on the stakeholder's goals and what impacts the uncertainty would have on the satisfaction of goals at design time. The authors in [27] apply decision analysis and multi-objective optimisation techniques based on Monte-Carlo simulation to assess the consequences of uncertainty and therefore identify the potential risks. Although, all the above approaches support the SAS in dealing with uncertainty, none

of these approaches supports the quantification of the level of uncertainty. In contrast, our proposed approach supports the modelling of the uncertainty levels by classification of surprise, and thereby helps in risk assessment for SAS.

VIII. Conclusion and Future Work

This paper had as its goal the creation a framework to model uncertainty for NFRs of SAS and that could further be used to model risks for requirements, in that the paper has been successful. For the purpose of modelling the level of uncertainty we performed the classification of surprise to measure the size of surprise. Furthermore, multiple techniques have been demonstrated such as the calculation of failure rates that have allowed different amounts of risk to be calculated. The results of these techniques have then been used to classify different surprise groups based on the risk they indicate towards the functionality of the NFRs. The last major contribution was the study into the use of CCS and Bayesian surprise, the results of which were that use of CCS is circumstantial, though in the environment in may prove more useful than Bayesian surprise this was not always the case, even when performing the same task, whether or not it is used would ultimately come down to how important confidence is to the current task.

There are many areas that could be explored next based on the work in this paper. The first are studies into additional surprise definitions. It has been shown that CCS can be as good if not better than Bayesian surprise in modelling risk for this task, there are of course many more definitions of surprise, a comprehensive study could be done focusing on when and where these different surprise values should be used based on data in using similar SAS. Related to this is an expansion of the work in this paper into different types of SAS, it has now been seen what the results of this framework has been on RDMSim, though are many more types of SAS and though the RDMSim can be generalised to an extent more comprehensive study could still garner further insight into the works of this paper. Lastly, more unique and novel techniques may be explored in order to identify and model risk, this paper has proposed several, though there are no doubt many more that could be developed and assist designers of SAS.

Acknowledgment

This work has been part supported by the EPSRC Project Twenty20Insight (Grant No. EP/T017627/1) and EPSRC Project MDE-Net (Grant No. EP/T030747/1): sub Project1 ”ReqModAI Requirements Models for Artificial Intelligence: Framework and Case Study” and sub Project2 ”iDecide: Quantifying Uncertainty in Models using Artificial Intelligence for Personalised and Shared Decision-Making in Digital Health”.

References

[1] M. Faraji, “Learning with surprise theory and applications,” 2016.

[2] P. Baldi and L. Itti, “Of bits and wows: A bayesian theory of surprise with applications to attention,” *Neural Networks*, vol. 23, no. 5, pp. 649–666, 2010.

[3] A. Modirshanechi, J. Brea, and W. Gerstner, “A taxonomy of surprise definitions,” *Journal of Mathematical Psychology*, vol. 110, p. 102712, 2022.

[4] N. Bencomo, “Quantun: Quantification of uncertainty for the reassessment of requirements,” in *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, pp. 236–240, IEEE, 2015.

[5] N. Bencomo and A. BELAGGOUN, “A world full of surprises: Bayesian theory of surprise to quantify degrees of uncertainty,” *36th International Conference on Software Engineering, ICSE Companion 2014 - Proceedings*, 05 2014.

[6] P. e. a. Sawyer, “Requirements-aware systems: A research agenda for re for self-adaptive systems,” in *18th IEEE International RE Conference*, pp. 95–103, IEEE, 2010.

[7] K. Welsh, P. Sawyer, and N. Bencomo, “Towards requirements aware systems: Run-time resolution of design-time assumptions,” in *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, pp. 560–563, IEEE, 2011.

[8] S. M. Hezavehi, D. Weyns, P. Avgeriou, R. Calinescu, R. Mirandola, and D. Perez-Palacin, “Uncertainty in self-adaptive systems: A research community perspective,” *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 15, no. 4, pp. 1–36, 2021.

[9] N. Esfahani, E. Kouroshfar, and S. Malek, “Taming uncertainty in self-adaptive software,” in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pp. 234–244, 2011.

[10] B. H. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, “Software engineering for self-adaptive systems. lncs, vol. 5525,” 2009.

[11] S. Hassan, N. Bencomo, and R. Bahsoon, “Minimizing nasty surprises with better informed decision-making in self-adaptive systems,” in *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 134–145, IEEE, 2015.

[12] H. Samin, L. H. G. Paucar, N. Bencomo, C. M. C. Hurtado, and E. M. Fredericks, “Rdmsim: An exemplar for evaluation and comparison of decision-making techniques for self-adaptation,” 2021.

[13] D. Weyns, “Software engineering of self-adaptive systems,” *Handbook of software engineering*, pp. 399–443, 2019.

[14] K. Keeton, C. A. Santos, D. Beyer, J. S. Chase, J. Wilkes, *et al.*, “Designing for disasters.,” in *FAST*, vol. 4, pp. 59–62, 2004.

[15] M. Ji, A. C. Veitch, J. Wilkes, *et al.*, “Seneca: remote mirroring done write.,” in *USENIX Annual Technical Conference, General Track*, pp. 253–268, 2003.

[16] Z. Chen and F. Cao, “The approximation operators with sigmoidal functions,” *Computers & Mathematics with Applications*, vol. 58, no. 4, pp. 758–765, 2009.

[17] https://github.com/TheSequel02/Modelling-Uncertainty-for-Requirements-The-case-of-Surprise-Results.

[18] G. Hamerly and C. Elkan, “Learning the k in k-means,” *Advances in neural information processing systems*, vol. 16, 2003.

[19] T. M. Mitchell *et al.*, *Machine learning*, vol. 1. McGraw-hill New York, 2007.

[20] H. Giese, N. Bencomo, L. Pasquale, A. J. Ramirez, P. Inverardi, S. Wätzoldt, and S. Clarke, “Living with Uncertainty in the Age of Runtime Models,” in *Models@run.time: Foundations, Applications, and Roadmaps*, Lecture Notes in Computer Science, pp. 47–100, Cham: Springer International Publishing, 2014.

[21] J. Y. Halpern, *Reasoning about uncertainty*. MIT press, 2017.

[22] L. H. G. Paucar and N. Bencomo, “Knowledge base k models to support trade-offs for self-adaptation using markov processes,” in *SASO*, pp. 11–16, IEEE, 2019.

[23] J. Whittle, P. Sawyer, N. Bencomo, B. H. Cheng, and J.-M. Bruel, “Relax: Incorporating uncertainty into the specification of self-adaptive systems,” in *RE*, IEEE, 2009.

[24] J. Whittle, P. Sawyer, N. Bencomo, B. H. Cheng, and J.-M. Bruel, “Relax: a language to address uncertainty in self-adaptive systems requirement,” *Requirements engineering*, vol. 15, pp. 177–196, 2010.

[25] A. J. Ramirez, B. H. Cheng, N. Bencomo, and P. Sawyer, “Relaxing claims: Coping with uncertainty while evaluating assumptions at run time,” in *MODELS*, Springer, 2012.

[26] M. Kwiatkowska, G. Norman, and D. Parker, “Probabilistic symbolic model checking with prism: A hybrid approach,” *International journal on software tools for technology transfer*, vol. 6, pp. 128–142, 2004.

[27] E. Letier, D. Stefan, and E. T. Barr, “Uncertainty, risk, and information value in software requirements and architecture,” in *ICSE*, 2014.

[28] E. Letier and A. Van Lamsweerde, “Reasoning about partial goal satisfaction for requirements and design engineering,” in *Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering*, pp. 53–62, 2004.