Minimizing Nasty Surprises with Better Informed Decision-Making in Self-Adaptive Systems

Sara Hassan*, Nelly Bencomo† and Rami Bahsoon*
*School of Computer Science
University of Birmingham, Birmingham, UK B15 2TT
†ALICE: Aston Lab for Intelligent Collectives Engineering, CS Group,
Aston University, Birmingham, UK
Email: ssh195@bham.ac.uk, r.bahsoon@cs.bham.ac.uk, nelly@acm.org

***Abstract*—Designers of self-adaptive systems often formulate adaptive design decisions, making unrealistic or myopic assumptions about the system's requirements and environment. The decisions taken during this formulation are crucial for satisfying requirements. In environments which are characterized by uncertainty and dynamism, deviation from these assumptions is the norm and may trigger "surprises". Our method allows designers to make explicit links between the possible emergence of surprises, risks and design trade-offs. The method can be used to explore the design decisions for self-adaptive systems and choose among decisions that better fulfil (or rather partially fulfil) non-functional requirements and address their trade-offs. The analysis can also provide designers with valuable input for refining the adaptation decisions to balance, for example, resilience (i.e. satisfiability of non-functional requirements and their trade-offs) and stability (i.e. minimizing the frequency of adaptation). The objective is to provide designers of self-adaptive systems with a basis for multi-dimensional what-if analysis to revise and improve the understanding of the environment and its effect on non-functional requirements and thereafter decision-making. We have applied the method to a wireless sensor network for flood prediction. The application shows that the method gives rise to questions that were not explicitly asked before at design-time and assists designers in the process of risk-aware, what-if and trade-off analysis.**

I. Introduction

Recently, research on self-adaptive systems has been primarily concerned with run-time solutions for self-adaptive decision-making and has not sufficiently discussed the role that off-line design and systematic evaluation can play in informing the problem. The need becomes crucial in systems which exhibit scale, dynamism, uncertainty and risks in the execution environment, where hidden and unexpected behaviours are the norm. The fundamental premise is that early exploration and "stress" evaluation of self-adaptive design space can uncover hidden behaviours, reveal likely risks and magnify some trade-offs. This exercise can provide better understanding of the requirements and more realistic assumptions of the environment. It can consequently feed into better formulation and elaboration of the self-adaptive design decisions, enhance our assumptions of the system and even support a more efficient decision-making mechanism at runtime. As a motivating example, consider self-adaptive multi-agents systems that operate in environments that are dynamic, stochastic, and partially observable. The agents collect information and revise their plans at run-time. Variables that describe the properties of the environment can take a range of continuous values. Run-time decisions made by the agent depends on the belief state at each point in time, i.e. the posterior probability distribution over all possible states, given all evidence to date [1]. The agent often uses little evidence to update its beliefs about the likelihood of all the possible future states. Certain types of agents make use of evidence provided to them from the environment through sensor models. They can revise their plans to handle further evidence [2]. They can exhibit graceful degradation under time pressure and in complex environments [2]. This situation can be prevented with better design time decision support to probe for risks in the design space and formulate self-adaptive decisions that can better work under uncertainty.

Our previous work on decision-making for self-adaption under uncertainty has been primarily concerned with online decision problem [3], [4]. That contribution was concerned with adaptation of the system's behaviour at run-time in response to changes in the operational requirements and the environment. Our contribution here however, aims at supporting the designers of self-adaptive systems at design time.

We posit that designers of self-adaptive systems often formulate design decisions making unrealistic assumptions about the system's requirements and environment. In environments which are characterised by uncertainty and dynamism, deviation from these assumptions are common and may trigger "surprises". A surprise measures how an observed event affects assumptions [3]–[5]. Specifically, in this context a surprise is defined as the difference between the belief distributions of the designer prior to and posterior to a given event regarding the need to trigger adaptations when changes occur in the execution environment. We propose a systematic method which makes the link between the emergence of surprises, risks and design trade-offs explicit. The method can be used to explore the design decisions of self-adaptive systems and choose among decisions that better (and perhaps partially) fulfil non-functional requirements (NFRs) and their trade-offs. We specify partial fulfiment here due to the vagueness of NFR formulation [6]; partial fulfilment is referred to from hereon as "satisficement".

The method searches for surprises through "exploratory analysis"; it probes for large range of values for a monitorable, which exceeds designer's expectation of the execution environment, to reveal new and hidden behaviours. As surprises can manifest into risks, we devise a multi-attribute decision-making mechanism to evaluate the significance of

the surprises (along with other likely risks) and, also, the impact of those risks. The mechanism quantifies the added value of the self-adaptive design decisions in satisfying NFRs and their potentials in mitigating risks (including surprises). The mechanism draws inspiration from the Security Attribute Evaluation Method [7] and includes two stages: (1) multi-attribute risk assessment of the consequence of surprises (and other risks) and (2) multi-attribute benefit assessment of the self-adaptive design decisions in mitigating risks. Pareto analysis is used to explore the solution space, represented by design decisions. Specifically, the analysis looks for a decision that satisfices NFRs and their trade-offs. The overall objective of the method is to provide designers of self-adaptive systems with a basis for multi-dimensional what-if analysis. The method can provide insights into formulating and elaborating adaptive design decisions. It can also inform whether (i) the benefits of triggering adaptations are likely to outweigh the cost/overheads; (ii) partial satisficement of the NFRs can be tolerated, (iii) adaptations are unavoidable and/or (iv) the need for new adaptive decisions or refining/elaborating existing one. We have applied the method to a wireless sensor network for flood prediction. The application shows that the method gives rise to questions that were not explicitly asked before at design-time and assist designers in risk-aware what-if and trade-off analysis.

The remainder of this paper is structured as follows: Section II elaborates on the quantifications needed for the method and explains briefly the techniques used to provide those quantifications. Section III presents the method. Section IV applies the method to a case study and V discusses its application. Section VI presents the related work. Section VII concludes and discusses future work.

II. Concepts to Support Decision Making

This section presents the concepts needed by the method to quantify the effect of partially anticipated execution environments on the satisficement of NFRs and analyze the effect of alternative design decisions on the NFRs considered. Section II-A sets the context for the method presenting the quantifications needed in order to address the offline decision problem of deciding the initial behaviour of a self-adaptive system. Section II-B presents existing concepts which will be used to quantify the effects named above.

A. Needed Quantifications

A way to quantify the impact of different execution environments on the level of satisficement of NFRs is needed. This is due to the need of analyzing how different execution environments will affect the behaviour of the self-adaptive system and therefore, the need for triggering adaptations. On the other hand, a threshold is required to identify when an adaptation needs to be triggered in the self-adaptive system. In addition, techniques are needed to quantify the effectiveness of different adaptation design decisions on staying below this threshold. They will assist the self-adaptive system designers in deciding which decision is optimal in terms of satisficing conflicting NFRs while aiming to continue satisfying the NFRs without the need to trigger an adaptation. Finally, a method to integrate these quantities into one representation across several NFRs is needed for trading-off analysis across different NFRs.

In essence, the method presented in this paper aims to balance the satisficement of conflicting NFRs as well as assisting the designer of the self-adaptive system in defining better design decisions. Also, design-time analysis, as it is presented here, is an exploratory approach compared to run-time analysis. In other words, the questions that arise while conducting this method would not otherwise come to light if the focus had only been on the run-time analysis of the self-adaptive system. The exploratory approach gives the designers of the self-adaptive system as well as the stakeholders a deeper understanding of the expected behaviour of the system before it is deployed. This justifies the overhead of combining several quantities in a single method.

B. Available Quantification Techniques

Given the needed quantifications identified above, this section presents concepts available in the literature that could be used to represent the quantities needed. First, the concepts are presented separately. Then, their combination into a single systematic method is presented in the following section.

1) Bayesian Surprise: The concept of Bayesian surprise is defined as a measure of how observed data affects the models or assumptions of the world during runtime [5, p. 1]. The unit of measurement of surprises is "*wow*". The "*wow*" unit permits the quantification of how much beliefs increase or decrease according to observed data. A large surprise value means that the evidence provided from the environment has caused a large difference between the prior and posterior probabilities of an event. Lets us have a non-functional requirement NFR_i, and D representing the evidence provided by the properties monitored as variables in the execution environment (from hereon called monitorables). $P(NFR_i)$ is the prior probability of the non-functional requirement NFR_i being partially satisficed and $P(NFR_i|D)$ is the posterior probability of the NFR_i being partially satisficed given the evidence D. The surprise estimates the divergence between the prior and posterior distributions and is calculated by using the Kullback-Leibler divergence (KL) [8]:

$$S(NFR_i, D) = KL(P(NFR_i|D), P(NFR_i)) = \sum_i P(NFR_i|D) \log \frac{P(NFR_i|D)}{P(NFR_i)} \quad (1)$$

The initial values of $P(NFR_i|D)$ and $P(NFR_i)$ are given as an input to the approach from stakeholders. In [5], the system accumulates knowledge at runtime about the environment. The assumption made is that the greater the magnitude of the surprise value, the higher the likelihood that an adaptation in the behaviour of the system needs to be triggered.

In [5], Bayesian surprises are exploited during runtime to support decision-making at runtime. The approach supports

the quantification of uncertainty over different time slices at runtime and helps the system improve its behaviour based on learning. The dynamic decision networks (DDN) approach developed in [3] makes use of the surprises concepts to reason about uncertainty in the decision-making problem over different time slices. However, this learning process is memory intensive and therefore, carrying out the analysis extensively would consume memory. In contrast, in the method presented here, and since it is a design-time method, instead of moving across time, only 1 time slice is used.

Our method aims to select an optimal design decision, with respect to balancing the satisficement of conflicting NFRs, at design time by:

- Assisting designers of self-adaptive systems in coming to an informed decision for the initial design of the self-adaptive system. It has to be noted that the decision made here will not necessarily remain the optimal decision throughout the life of the system, but it creates a more informed starting point.
- Formulating the range of identifiable execution environments and the effect they have on the satisficement of the NFRs.
- Highlighting the monitorables in the execution environment which have the highest effect on the satisficement of the NFRs.
- Creating a basis for multi-dimensional what-if analysis at design-time to assess the behaviour of the system in different independent execution environments. What-if analysis pushes the envelope and stresses the system. Even if some of the scenarios considered in the what-if analysis might not initially seem reasonable, at some point in the future these might be realistic requirements for changing the design of the system in response to changes in the execution environment. Stakeholders therefore might like to understand the ramifications of such changes [9].
- Creating a quantified documentation of the beliefs at design-time to be compared with the behaviour of the system later at run-time.

The main contribution of using the concept of surprises for the offline decision problem is the what-if analysis mentioned in the third point above. A Bayesian surprise in this method is used to revise the designers' initial understanding of the execution environment and the system's interaction it. This update happens by exploring the range of possible values associated with the monitorable rather than just the anticipated values at the time of system deployment. Because a large range of monitorables is analysed rather than just the anticipated values, the need for triggering adaptations over the long term can be analysed updating the belief of the stakeholders about the relatively long-term impact of the execution environment on the system's behaviour beyond the deployment time.

2) Multi-attribute analysis: "Multi-attribute analysis techniques help decision makers evaluate alternatives when conflicting objectives must be considered and balanced and when outcomes are uncertain" [10]. Multi-attribute analysis used in our method is done in two stages: (1) multi-attribute risk assessment and (2) multi-attribute benefit assessment. For multi-attribute risk assessment in relation to security, for example, threats are defined as events, such as denial of service attacks, procedural violations, IP spoofing, etc., which could lead to an information system compromise. Butler and Fischbeck [11] define the multi-attribute analysis as: an attack (a) is an instance of a threat that results in information of the system being compromised and produced an outcome (Oa) of one or more consequences (Xi). For example, compromising the system may ultimately result in lost revenue (X1), public embarrassment (X2), lost productivity (X3), and damaged corporate image (X4) [11, p. 3]. The impact of these threats can be quantified using the threat index [7]. The calculation of threat index considers the expected, high, and low outcomes possible for a threat in terms of their consequences (consequences hereby called risk outcome attributes), and the probability of each of this outcomes. The formula for calculating threat index is defined in [7]. Furthermore, benefit assessment can be carried out to assess the effectiveness of alternative tasks on reducing the threat index [7]. A novelty of the method presented here is that it allows the assessment of the effectiveness of the alternative decisions while optimising the behaviour of the system in such a way that it does not exceed a given surprise tolerance threshold.

3) Pareto Analysis: The original reason behind using the Pareto analysis in [12] was to develop an environment where defining the decisions that satisfy the functional requirements is controlled by the designer, while identifying decisions that satisfice the NFRs is left to the machine [12, p. 1]. The human designer specifies the NFRs which need to be satisficed, but an automated decision is made regarding a sub-optimal solution that can satisfice them. The method presented here uses a Pareto front to used to explore the solution space. The dimensions represent NFRs which trade-off each other. Each point on the Pareto front represents a possible design decision that satisfices each of the NFRs to some extent. The coordinates of each point represent contribution of each alternative decision to the satisficment of each NFR respectively. There is no limit on the number of NFRs (dimensions) used. The value of each coordinate is derived from the multi-attribute analysis phase.

III. Decision Making Method

This section presents the steps of the method. An overview of the method is illustrated in Fig. 1. For this method, we assume that NFRs and possible design decisions that satisfice them can be deduced from goal models (e.g. i* framework). NFRs are "fine-grained goals under the sole responsibility of the software-to-be [13, p. 1]."

A. *Exploratory Analysis to Detect Surprises*

*1) **Calculating Surprise Value Distributions**:* Designers have to agree on critical NFRs that need to be monitored to identify hidden and new surprises. The exploratory analysis to search surprises aims at stressing the system with a wide range of monitorable values. Consequently, this exercise can help in studying the potential that the design decisions withstand

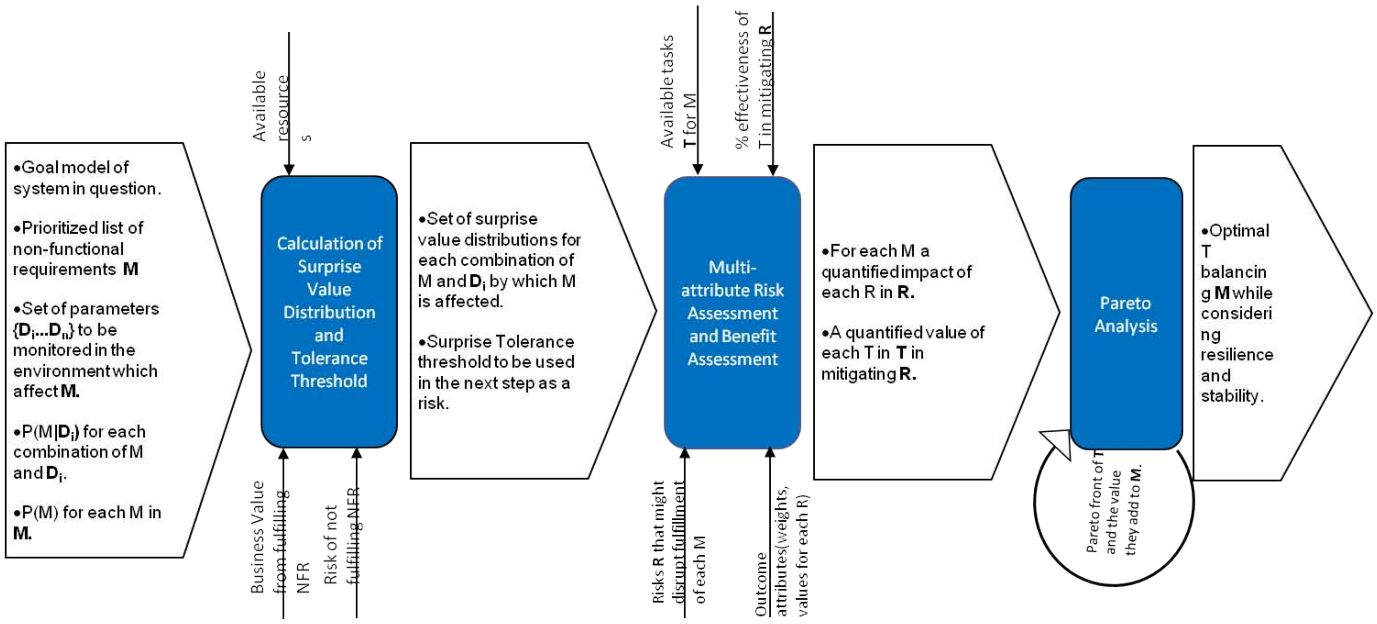

Fig. 1. Overview of the method. T signifies the alternative design decisions; M signifies each NFR; $\mathbf{D_1..D_n}$ signify the monitorables; **R** signifies the set of risks that can disrupt the NFR satisficement

these surprises and succeed in mitigating their risks. The involvement of various stakeholders and end users to decide on the range of inputs is highly desirable; users should be treated as collaborators in the design process rather than pure consumers of system's functionalities [14, p. 15]. The values of $P(NFR_i|D)$ and $P(NFR_i)$ are given as input to the method. Although outside the scope of the method, one possibileto derive the values of $P(NFR_i|D)$ and $P(NFR_i)$ is to use the Discrete Time Markrov chains [15]. It helps encode the NFRs and the domain knowledge about the system into states and then the PRISM probability checker is used to derive the probability that each NFR is satisfied. The $P(NFR_i|D)$ values form a probability distribution over the random variable D.

Each monitorable is linked to the list of NFRs which it affects. For each pair of non-functional requirements and monitorables (**D**), the surprise of associated with each expected value of the monitorable is calculated using: (1) D=value of monitorable, (2) $P(NFR_i|D)$= probability that NFR_i is satisficed given that value of D and (3) using the formula for calculating surprises in II-B. Repeating the calculation for some or all of the values that **D** can take will produce a distribution of surprise values for each pair of monitorable and NFR. The granularity of the distribution can be increased with iterations of this exercise to help the designer identify a design decision at the level of granularity s/he needs. Granularity refers to the number of monitorable values for which the surprise value was calculated explicity (the more values the more granular the distribution is).

To generate a surprise value distribution, the range of values for each monitorable is first used to generate the x-axis of the plot area. For monitorables that can be seen as continuous variables, "calibration" values are used instead to calculate the surprises (e.g. if the monitorable is response time in seconds and it can take the values 0-1000 seconds, surprise values are calculated for response times 0,100,200...1000 seconds respectively). In turn, the stakeholders would only need to identify the $P(NFR_i|D)$ given these monitorable values. The y-axis is generated ranging from 0 to the maximum surprise value calculated. The x coordinate of each value in the plot area is then taken as the value of D and y coordinate is the surprise value associated with that value of D.

*2) **Calculating Surprise Tolerance Threshold**:* The NFR of the highest priority is the one which the stakeholders will always thrive to satisfice by triggering adaptations. This can be reflected on the distributions of surprise values by setting a threshold on each distribution to signify the surprise tolerance level related to that NFR. The significance of the threshold is that it can also help designers in the refinement and elaboration of NFRs that need to be satisficed through adaptation. It informs scenarios where the existence of an obstacle in the environment makes triggering an adaptation unavoidable. Conversely, it can inform scenarios where goals can still be relaxed to avoid unnecessary adaptation. It can be argued that this tolerance threshold is useful either addressing the offline or online decision problems. However, only the offline problem is the focus of this paper.

Fig. 2 illustrates an example of this. To interpret what this threshold means, let us consider the point where the surprise value is 0.02986. Only if the surprise value associated with 3.06 minutes increases from 0.02986 to 0.0675 will the system's behaviour be adapted. Below that, it is not worth trying to reduce the time to restart after failure to less than

3.06 minutes. The exact threshold value depends on the range of surprise values generated for that specific pair of NFR and monitorable.

Furthermore, the reason behind using a percentage to set the threshold is that there might be more than one surprise value distribution associated with each NFR if it is affected by many monitorables. Setting a hard and fast value as the threshold across all the distributions that are associated with an NFR is not reasonable. Percentages prove to be more flexible in setting the threshold. Prioritizing the NFRs and thereafter coming up with the percentages which will make up the surprise threshold can itself prove to be a challenge. One technique for prioritization of NFRs is used in [3]. The business value of satisficing the NFR and the risk associated with not satisficing it are both to be considered as well as comparing the resources required for satisficing that NFR against the available resources. The possibility of further surprises being triggered due to a design decision choice should also be considered. A simplifying assumption made here is that the percentage of tolerated surprise should be no more than 50% of the overall range of surprise values of the distribution due to the overhead of triggering adaptations if the surprise tolerance is beyond that.

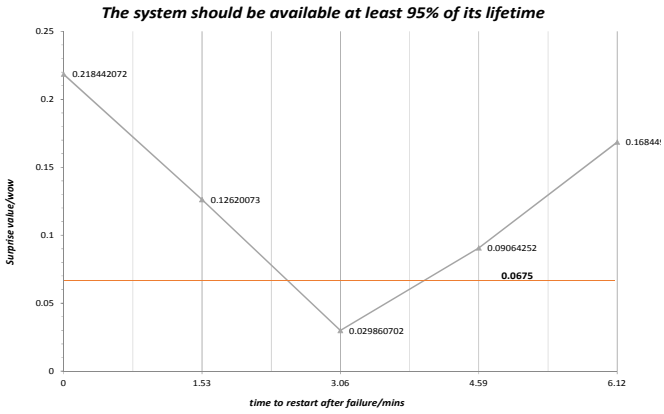

Fig. 2. Surprise value distribution against monitorable

B. Multi-attribute Analysis for Surprise Risks and Design Decisions Benefits

We devise a multi-attribute decision-making mechanism, which draws inspiration from the Security Attribute Evaluation Method (SAEM) [7] but abstracts it to any NFR. The mechanism encompasses two stages: (1) multi-attribute risk assessment of the consequence of risks that can disrupt the NFR satisficement and (2) multi-attribute benefit assessment of the self-adaptive design decisions in mitigating risks.

1) For each NFR, the stakeholders need to identify the risks that can hinder the satisficement of that requirement as well as the expected frequencies of each risk. Although the specific method for identifying risks is outside the scope of this method, we suggest the taxonomy-based questionnaire as a systematic technique for identifying risks [16]. The risk set of each NFR includes the risk of the surprise value exceeding its threshold (in the form "surprise value >x, when monitoring D" rather than "surprise value >x%"). This form of describing the risk related to the surprise tolerance threshold is needed to distinguish the pairs NFRs and monitorables from one another. Exceeding the surprise tolerance threshold is included as a risk because of the costs incurred in triggering adaptations if the surprise tolerance threshold is exceeded.
2) For each NFR, stakeholders need to identify the outcome attributes of the risks and then rank and weigh those attributes on a scale from 1-100, where 100 is the attribute that is most important to stakeholders. These weights are then normalized to a value between 0 and 1.
3) For each NFR, stakeholders identify the values of the outcome attributes for each risk associated with that NFR.
4) The values of the outcome attributes under each consequence need to be normalized by the designer of the system to one scale so that an overall threat index for each risk can be calculated. The method uses ideas from the AHP scale [17] which rely on stepwise comparison between alternatives using a 1-9 scale of relative importance. We compare the relative importance of the values of a certain outcome attribute under each risk.
5) The threat index of each risk is then calculated using the equation defined in [7]. After calculating the threat index of each threat, the summation of the threat indexes is also calculated.
6) For all the available design decisions, the % effectiveness of each alternative decision in reducing the risk is estimated by designers. It is not necessary that a design decision can reduce the threat index of all the risks. If design decisions can only reduce the threat index of a risk when combined, then the combination of the two decisions is considered as a separate design decision with its own % effectiveness.
7) For each design decision, the new threat index that each risk can achieve is calculated using the threat index values from Step 5 and the % effectiveness values from Step 6. The summation of these updated threat index values is then calculated to see the overall change that each decision can achieve. The % change in threat index which the decision can achieve is also calculated.

C. Pareto Analysis for Design Decisions & NFR Trade-offs

The previous phase can inform trade-off analysis between design decisions in relation to individual NFRs. However, NFRs exhibit conflicts and trade-offs among each other. Designers can decide to compromise the satisficement of some NFRs in order to satisfice others. Because of this trade-off between NFRs, Pareto analysis is included. One crucial condition for the Pareto analysis is ensuring that all design decisions that make up the Pareto front satisfy the functional requirements of the system (conformant analysis [12]). Each NFR in question forms a dimension for the space in which the Pareto front will be plotted. Each dimension has a scale from

0-100 which signifies the percentage effectiveness in reducing risks associated with that NFR. Each design decision is then plotted as a point, where every coordinate is the percentage effectiveness of the design decision in reducing the threat index in relation to the respective dimension. Thereafter, analyzing the Pareto front can lead to the discovery of *knee points*: a subset of design decisions where a small enhancement of one NFR can lead to a significant deterioration in another one.It is argued that finding the knee point corresponds to finding the optimal solution to the decision problem; the solution hopes to satisfy NFRs trade-offs [18]. An optimal solution in this context means a design decision that balances the satisficement of conflicting NFRs. A knee point is almost always the most preferred solution, since it requires an unfavourably large sacrifice in one objective to gain a small amount in the other objective. However, if a single knee point does not exist, then an area of optimal solutions is called the knee region. There are many techniques for identifying the knee-point on a Pareto front [18]. Iterations of Pareto analysis can be carried out to explore the design space refining the design decision choice. The significance of carrying out Pareto analysis as the last step of the method is carrying over the knowledge the likelihood of adaptations being triggered from the first phase of the method. Otherwise the plotted points on the Pareto front would have been an overestimation the added value of each design decision along each dimension.

In the next section, an application of the method on a self-adaptive system is presented using hypothetical data. In particular, the quantified NFRs, P($\mathbf{NFR_i}$), P($NFR_I\mathbf{D}$), risks associated with each NFR, the outcome attributes, their weights, values, and the percentage effectiveness of each design decision in mitigating the risk are all inspired by empirical data provided by a prototype of the case study.

IV. Case Study

A. GridStix

"The GridStix system is a wireless sensor network (WSN) for detecting and predicting flooding" [19]. Our choice for the case stems from the fact that its design space for self-adaptivity exhibits adequate level of complexity. It is characterized by: (i) numerous conflicting NFRs, (ii) vast possibilities of adaptation design decisions and (iii) execution environments which can be represented by monitorables. Specifically, the design space contains alternative design decisions that address an NFR at the cost of another one, which is needed for illustration of the method. The case illustrates how the method can be systematically applied and better informs design decisions for self-adaptivity in software systems. The fact that the case study is very well documented [19] allows us to highlight and discuss the benefits and improvements which are outcomes of our method. The latter was an extra reason to chose GridStix as the case study.

In a nutshell, GridStix provides: (i) "a resilient and adaptive sensor network, which collects information from a range of sensors and transmits the collated data off-site where it is used to perform spatial flood prediction [20, p. 4]", (ii) a light-weight computational Grid which performs local point-based flood prediction and calculation of flow rates from images [20] and (iii) a flood warning tool for local stakeholders, such as residents and businesses [20]. Adaptation to changing execution environments in GridStix relies on the use of instances of overlay component frameworks. Instances are made up of the overlay plug-ins which are the per-node implementations of network overlays [19]. Adaptation provides a base for building application-level networks (or overlays), which are typically used to implement new networking services not provided by the underlying network

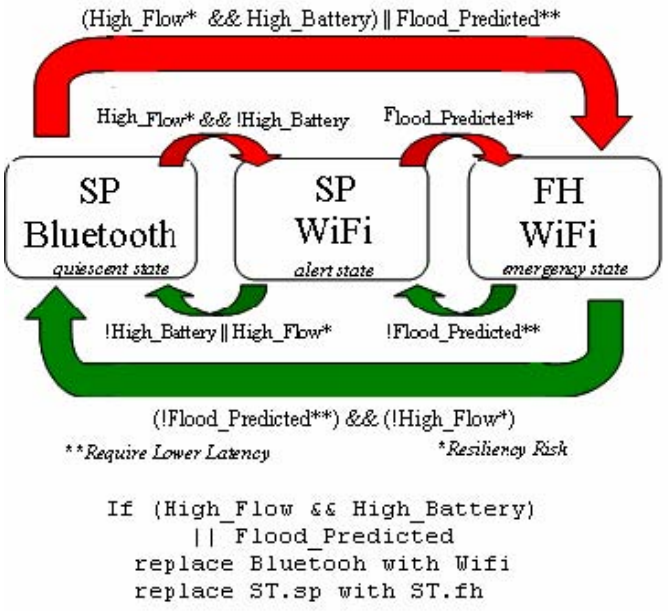

Fig. 3. General behaviour of GridStix [19]

Experts have defined three main "domains" of river behaviour: quiescent (most common), high flow and flooding [21] (Fig. 3). Each domain can be seen as a different state of the execution environment which requires a different behaviour from the self-adaptive system. Design decisions for GridStix are designed to take into account the NFRs, their trade-offs, their satisficement, suprise tolerance thresholds and monitorables for those 3 domains.

Goal models are used to specify and monitor NFRs. The NFRs in question are: (1) *Average power consumption per 1KB of data from node to gateway must be <1500 mW (refining the energy efficiency soft goal)* and (2) *Number of routes to the gateway must be >12 (refining the fault tolerance soft goal)*. For simplicity, only river depth is used as the monitorable to indicate the domain; it is measured using sensor nodes deployed in the river. The alternative design decisions can comprise one or more of the following choices:

1) *BlueTooth BT (for communication between sensor nodes along the river)*
2) *WiFi (for communication between sensor nodes along the river)*

3) *Shortest-Path SP spanning tree algorithm (this is used to search for the best path to transmit sensor readings)*
4) *First-Hop spanning FH tree algorithm (this is used to search for the best path to transmit sensor readings*
5) *Designing an alternative instance of the overlay component framework*
6) *Designing an alternative plug-in component*
7) *Adaptive design alternative with WiFi and SP*
8) *Adaptive design alternative with WiFi and FH*
9) *Adaptive design alternative with BT and SP*
10) *Adaptive design alternative with BT and FH*

B. Application

The analysis of the case is based on empirical data collected from a prototype of GridStix that employs 14 sensor nodes; deployed at River Ribble in the North West England. Recorded river depth values were (1) 1.14-2.25 m for the "quiescent" domian and (2) >4.34 m for the "flooding" domain [22]. The graphs in the [21] are used to define the NFRs and infer the P($\mathbf{NFR_i}$) and P($NFR_i|\mathbf{D}$) values. The claim refinement model in [21] is used to choose the monitorable. The risks, risk outcome attributes and risk outcome attribute values are then inferred from the explanation of the GridStix framework in [19], [20].

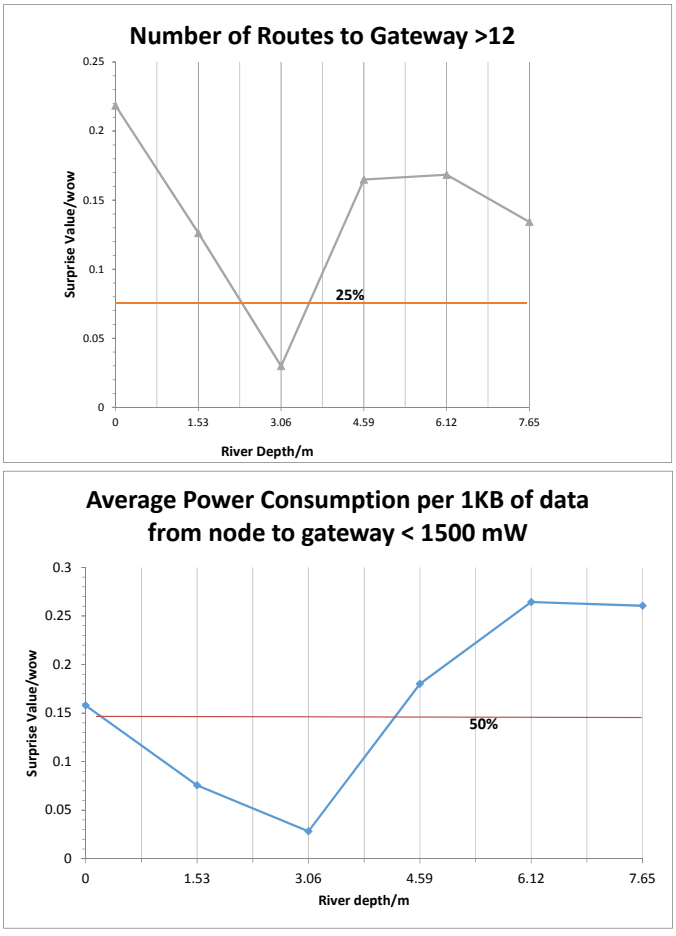

Fig. 4. Surprise value distributions and tolerance thresholds

*1) **Exploratory Analysis for Detecting Surprises**:* The distributions of surprise values in Fig. 4 assume that the soft goal *fault tolerance* – operationalized by the NFR *Number of routes to the gateway >12* – is of the highest priority. The surprise tolerance threshold for this NFR is 25% of the range of surprise values, as opposed to a threshold of 50% for the other NFR. This is a drastically simple way of calculating the surprise tolerance threshold – assuming there are only two NFRs and not considering any external factors. The distributions illustrate the effect of evidence has on the system's belief that the NFR can be satisficed. The lowest points in the graphs are when the river is 3.06 m deep; this depth is closest to the median of the quiescent "domain" river depth range. Since the quiescent "domain" represents the most common river behaviour, river depths in this domain do not provide further evidence to the system about the current execution environment and thereby the prior belief that the NFR will be satisficed does not differ much (i.e. the surprise value is low). River depths outside the quiescent "domain" present higher surprise values; they indicate a transition from one execution environment to another which would possibly trigger an adaptation depending on the surprise tolerance threshold. Surprise values that have more than 1 river depth associated with them signify that it is only the magnitude of the surprise that matters. Calculating surprise values for the river depths greater than 1.53 m yields a negative surprise value. However, as the magnitude of surprise matters the most, the sign is ignored. The change in surprise values is said to be significant only when the change is such that it exceeds its tolerance threshold. Such information is useful to planning alternative adaptation decisions in an attempt to satisfice the conflicting NFR(s). It is worth noting that due to the use of logarithm in the calculation of surprises, using $P(NFR_i|D)=0$ or $P(NFR_i)=0$ will result in an undefined value of surprise which cannot be plotted. The significance is attributed to the fact that a designer can embrace the uncertainty in the expected execution environments and design decisions. In particular, this can feed into eliciting possible causes (i.e. obstacles [23]) of the NFR disruption and re/thinking of alternative design decisions, which can cope with the risks/extreme cases. Alternatively, if the NFR has proven to be difficult to satisfice at design time, it is then imperative for the designers to use this information to further refine and elaborate the NFRs to improve the likelihood of their satisficement. Again, obstacle handling and resolution can be adopted to systematically refine and elaborate NFRs which are most likely hard to satisfice through goal relaxation, splitting, substitution, etc. [23].

*2) **Multi-attribute Analysis for Surprise Risks and Design Decisions Benefits**:* Figures 5 and 6 show part of the risk assessment and corresponding benefit assessment process for choosing the most effective adaptation design decision for the NFR *Number of routes to the gateway>12*. To illustrate what the numbers mean, the risk of exceeding the tolerance threshold is used as an example. The expected impact of this risk 0.167 delay in prediction and 0.142 regulatory penalties. The impact values do not have units in this case since

they have been normalized using the AHP scale for relative importance. The overall threat index of this risk is 38.302 and if WiFi is used as a design decision it is expected to reduce the threat index of that risk to 13.02. Furthermore, WiFi can reduce the threat index of risk 2 (which is not shown in Fig. 5) to 0 and reduce the threat index of the third risk from 24.925 to 19.94. Overall, WiFi can reduce the total threat index by 40.29%. The risk outcome attributes used in the risk assessment process are specific to each NFR. The risk of the surprise value exceeding the threshold would have the highest impact on the satisficement of that NFR since it has the largest threat index value (according to the calculations over all the risks not shown in the figure). Such information would entail the design of alternative adaptation design decisions, which minimize the impact of fragmentation in the sensor network. Looking at the benefit assessment table, the design decision with the highest effectiveness in mitigating the risks related to the NFR is that of WiFi for communication between the sensor nodes. In other words, if the designer wishes to optimize on that NFR, s/he would use a design decision which uses this communication medium in the design of GridStix. As it is inevitable that NFRs conflict, we use Pareto analysis in the next step.

Number of routes to gateway > 12 – Risk assessment				
Threat	Consequence	Regulatory penalties (0-6 scale)	Delay in prediction (Hrs)	Threat Index
1) Surprise value > 0.077 (219/yr)	Expected	0.142	0.167	38.302
	High	0.714	0.737	
	Low	0.142	0.094	
3) The constraint violates external robustness regulations enforced on the business (127/yr)	Expected	0.239	0.0909	24.925
	High	0.623	0.818	
	Low	0.137	0.0909	

Fig. 5. Fault tolerance risk assessment

Threat	WiFi	BlueTooth	WiFi + SP	WiFi + FH	BlueTooth + SP
1	13.02	38.30	32.67	0	38.30
2	0	8.55	7.29	0	8.55
3	19.94	24.92	24.92	14.95	24.92
Change (in %)	40.29	0	8.31	60.28	0

Fig. 6. Fault tolerance benefit assessment

*3) **Pareto Analysis for Design Decisions and NFR Trade-offs**:* Fig. 7 shows the application of Pareto analysis to the GridStix case. The graph produced is sparse due to the simplicity of the illustrative example. Its complexity can

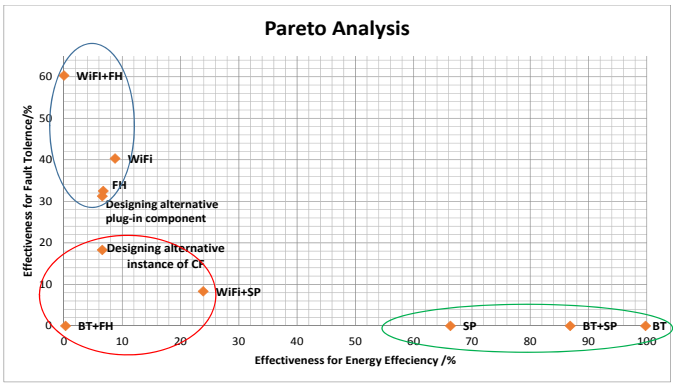

Fig. 7. Choosing the optimal task using Pareto analysis

grow with the design alternatives and dimensions (i.e. NFRs) under analysis. The points on this graph can be split into 3 regions (forming the 3 circles in Fig. 7). If the designer wishes to favour the satisficement of *Number of routes to the gateway >12*, then the designer examines the decisions within the blue circle. Iterations of Pareto analysis can be carried out exploring refined alternative design decisions within that region (e.g. exploring different bandwidths of WiFi or looking at specific functionalities to be incorporated in alternative plug-in components). If the designer wishes to favour the satisficement of *Average power consumption per 1KB of data from node to gateway <1500 mW*, then the decisions in the green circle would be more favourable option. Finally, to get a moderate satisficement for both NFRs, the design decisions in the red circle are considered. The choice of the region to consider depends on the prioritization of the NFRs and the expected execution environment at deployment time.

V. Discussion and Future Work

The method provides systematic guidance for designers of self-adaptive systems through a set of self-contained steps. It makes explicit the link between surprises, adaptation design decisions and risk. Crucially, it puts trade-offs at the heart of the decision-making process. This is relevant as the method could be used, for example, to balance resilience (i.e. the extent to which NFRs are satisficed) with respect to stability in self-adaptive systems (i.e. when to avoid unnecessary adaptation versus considering a necessary one). The designers of GridStix would have initially chosen design decisions which use BT and SP, assuming that the initial state of the river behaviour would be quiescent. Instead, we argue that a better choice can involve decisions which use WiFi and SP since it provides a balance between the two conflicting NFRs. In particular, plotting surprise value distributions gives rise to situations in the execution environment that the designers have not been considered before. Including a surprise tolerance threshold in the analysis can inform about situations where (i) unnecessary adaptations can potentially be avoided and (ii) trade-off between NFRs could be further elaborated, refined or relaxed to stabilize the system while having resilience qualified as"good enough". In GridStix, for example, *energy efficiency*

was a candidate to be relaxed to stabilize the system while satisficement was achieved. Including the multi-attribute risk analysis serves the objective of quantifying the impact of risk and added value of the available design decisions on more than one dimension. Pareto analysis provides a powerful tool to visualize the solution space and the contribution of design decisions to NFR trade-offs. Iteration of this Pareto analysis can lead to more refined decisions.

There are however a number of practical challenges involved in applying the method. We present these challenges below and suggest possible solutions to them that can be integrated in the future with the presented method.

1) **Validity of the choice made**: In the original work [21], running a simulator of GridStix falsified claims that BT was too risky to use due to its low benefit to fault tolerance. The falsified claims were (1) the mean number of nodes reachable with BT <WiFi and (2) data collected over >n hours is less in BT than in WiFi. The experiment in [21] concluded that BT had higher net impact on energy efficiency and fault tolerance than WiFi. The results of applying the method here are not consistent with this conclusion, although they are consistent with the original intuition of the strengths and weaknesses of the design decisions [19]. The difference between the conclusion of [21] and the conclusion of this method is due to the different views of the analysis. The method presented here is not relying on running a simulation of GridStix, which makes the conclusions present in the method arguable compared to [21]. Ultimately, at least running a simulator or prototype of GridStix can help validate the results presented here against [21]. However, by applying the method to the case study, the results can be used to update the stakeholders' beliefs about the system's run-time behaviour. After all, the satisifiability of the NFRs given the chosen design decision can only be tested at run-time. Furthermore, the extent to which the chosen design decision balances between conflicting NFRs' satisficement can only be shown at run-time.
2) **Scalability of the analysis**: The scalability of the analysis to a large number of NFRs and monitorables tend to be limited in the absence of dedicated tools for processing the information gathered from experts or historical data which is required for applying the method. Moreover, including more dimensions (i.e. NFRs) in the Pareto analysis can complicate that phase by producing a Pareto front that is too hard to visualize. However, pragmatic solutions exist to deal with this problem including the use of projection, graphics or virtual reality solutions to help visualize Pareto fronts and identify the knee-point or knee region [12]. These solutions are desirable in scenarios exceeding 3D space (i.e. 3 conflicting NFRs).
3) **Visualization of the results**: It is acknowledged that "human fatigue" is a common problem in methods involving Pareto analysis with a large number of points on the Pareto front. The problem needs smart selection and presentation techniques that can quickly and unobtrusively extract human assessments [12]. The use of automated pattern analysis and eye-tracking techniques have been cited as one solution to the problem [12].
4) **Stakeholder identification and reliability of inputs**: Ensuring the reliability and completeness of the inputs to the method is a prerequisite for meaningful analysis and has been cited as an inherent challenge for human-centred analysis methods. Perfect knowledge, sound judgement, and a consensus of opinion among the considered stakeholders are not realistic to assume. However, the Software Engineering literature provides pragmatic solutions and systematic approaches that, if adopted, can minimize the effect of this problem. Contributions including stakeholders identification (e.g. Stakenet [24]), multi-perspective analysis and negotiation [25] and classical architectural evaluation methods (e.g. ATAM [9]) are among the few approaches that can further improve our method.
5) **Balancing resilience and stability**: Resilience comprises two key attributes: dependability and robustness. Dependability is "the ability to deliver service that can justifiably be trusted [26, p. 13]" despite continuous changes. Robustness refers to the ability of the system "to deliver a service in conditions which are beyond its normal domain of operation [27]." Resilience is therefore the persistence of service delivery that can justifiably be trusted when facing changes. Persistence is determined by the capability of (effective) self-adaptation and the impact of changes on the service provided (from the user's perspective) [28]. Stability, however, is concerned not as much with the quality of the performance of the system but rather with the consistent non-variable performance in the system. In self-adaptive systems, one of the critical measures for stability is the frequency of adaptations. Reducing the frequency of unnecessary adaptations can lead to more stable systems. Interestingly, a system can be resilient and still fluctuate greatly. Moreover, low stability can be beneficial for continuously appraising resilience. An open question is whether the decisions provided by this method can "sustain" balancing between resilience, stability, conflicting NFRs and mitigating residual risk. Dynamic search-based techniques can render neat solutions to this; and therefore it is a subject of future work.

VI. Related Work

The related work presented here is divided into 2 categories: (A) work done on quantifying the uncertainty in design decisions and (B) work done on identifying the added value of alternative design decisions.

A. Quantifying Uncertainty in Design Decisions

Different techniques have been used to quantify the uncertainty of a software decision. Leiter et al. address uncertainty with the following question: "If, before making a decision, decision makers could pay someone to obtain additional information that reduce uncertainty about the cost and benefits of alternatives, how much would that information be worth to them?" [29]. The quantified answer to this question, called the expected value of perfect information (EVPI), helps decision makers focus on reducing uncertainty in decisions where

reducing uncertainty would actually be worthwhile. While EVPI helps decision makers define the aspects of the system which are worth focusing on in terms of reducing uncertainty, the use of surprise values in this method helps assess the effect of information on the need to make a decision (i.e. trigger an adaptation in the system's behaviour).

RELAX [30] enables developers to identify uncertainty in the requirements, thereby facilitating the design of systems that are, by definition, more flexible and amenable to adaptation in a systematic fashion. RELAXation modifiers like "eventually" or "until" are used to specify uncertainty in dynamically changeable requirements. A relevant possible avenue for future work is the use of the surprise tolerance threshold as suggested thresholds for RELAXation of NFRs at runtime.

B. Identifying Added Value of Alternative Design Decisions

Several approaches have been developed to manage the decision-making process in self-adaptive systems. Elkhodary et al. [31] developed the feature-based FUSION framework in which the engineers' knowledge of the self-adaptive system is represented in a model that classifies the possible configurations of the system according to validity and practicality. In contrast, the method presented here uses Pareto analysis as a top-down approach to making a decision, reifying and refining it with each iteration of Pareto analysis. This approach is more flexible than FUSION since it gives designers of the system the freedom to choose the level of detail of the analysis done.

Further away from self-adaptive systems is the net option value (NOV) approach developed in Sullivan et al.'s [32] approach to evaluate software modularity. Similar to the idea of EVPI, each alternative in this approach creates an option in the sense that it gives designers the right to invest in searching for a better alternative and replacing the current choice or deciding to keep it. This approach is tailored to the fine-grained software modularity design problem. The equivalent of this in the method presented here would be an iteration of the Pareto analysis where the design space is more refined. The decision problem addressed by this method looks at a more general decision problem where the design space is still to be explored.

The following text summarizes the delta with respect to the reviewed literature:

- Our method is more comprehensive in terms of the variety of questions it raises. Previous decision-making techniques address fewer aspects of the decison problem [29], [32].
- The aspects addressed are formulated flexibly in this method (e.g. NFRs). Previous work relies on a more rigid formulation of the design space (e.g. encoding the designers knowledge as features in FUSION [31], creating the design structure matrix to formulate the NFR constraints in NOV [32]).
- One aspect the method addresses is the issue of conflicting NFRs and the possibility to optimize the resilience and stability of the self-adaptive system at run-time. This has not been addressed in the reviewed literature.
- The method introduces the possibility to revisit NFRs through the concept of the surprise tolerance threshold aiming to optimize the stability of the system at run-time. This has not been addressed before in the context of self-adaptive systems.
- Fundementally, we address the design-time problem as opposed to runtime decision-making techniques where knowledge about the environment can be accumulated (e.g. [3]).

VII. Conclusion

We have contributed a systematic method – which makes the link between the emergence of surprises, risks and design trade-offs explicit – by combining techniques from surprise values, multi-attribute risk analysis and Pareto fronts analysis. The method explores the design time space of self-adaptive systems and searches for surprises. It probes for a large range of surprise values which stress the design to reveal new and hidden behaviours. A prioritized list of the NFRs which the system needs to satisfice is among the inputs. The expected execution environment properties are modelled as variables (monitorables) that can take a range of values. Surprise value distributions are used to assess the impact of the monitorable on the satisficement of NFRs. Observations of this step can feed into the design of new adaptive strategies or it can inform the elaboration of design decisions given the different values of the monitorables. The method uses a multi-attribute decision-making mechanism to evaluate the significance of the surprises (along with other risks) and magnitude of the impact they have. In addition, the mechanism quantifies the added value of the self-adaptive design decisions in satisfying NFRs and their potentials in mitigating risks (including surprises). The mechanism encompasses (1) multi-attribute risk assessment of the consequence of the surprises and (2) multi-attribute benefit assessment of the self-adaptive design decisions in mitigating risks. Pareto analysis is used to explore the solution space, represented by design decisions, aiming to find a design decision which satisfices NFRs and addresses their trade-offs. The use of Pareto front can assist in identifying decisions which balance between satisficing conflicting NFRs. The analysis can be carried out iteratively to come to a more refined decision. The analysis can serve the fundamentals of balancing between resilience and reducing the overheads of unnecessary adaptations. We have applied the method to a wireless sensor network for flood prediction. The analysis of the case is based on empirical data collected from a prototype of GridStix that employs sensor nodes realistically deployed at River Ribble in the North West England. The application shows that the method gives rise to questions that were not explicitly asked before at design-time and assist designers in risk-aware what-if and trade-off analysis. The method can aid designers of self-adaptive systems in performing design-time what-if analysis. The method can provide insights into formulating and elaborating adaptive design decisions. It can also inform whether (i) the benefits of triggering adaptation are likely to outweigh the cost/overheads; (ii) partial satisficement of the NFRs can be tolerated, (iii) adaptation is needed or can be unavoidable and/or (iv) the need for new adaptive decisions or refining/elaborating existing ones.

REFERENCES

[1] S. Russell, "Learning agents for uncertain environments (extended abstract)," in *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, ser. COLT' 98. New York, NY, USA: ACM, 1998, pp. 101–103. [Online]. Available: http://doi.acm.org/10.1145/279943.279964

[2] S. Russell and P. Norvig, *Artificial intelligence: A modern approach*, 2nd ed., ser. Prentice Hall series in artificial intelligence. Prentice Hall, 2003.

[3] N. Bencomo and A. Belaggoun, "Supporting decision-making for self-adaptive systems: From goal models to dynamic decision networks," in *Requirements Engineering: Foundation for Software Quality*, ser. Lecture Notes in Computer Science, J. Doerr and A. Opdahl, Eds. Springer Berlin Heidelberg, 2013, vol. 7830, pp. 221–236.

[4] N. Bencomo, A. Belaggoun, and V. Issarny, "Dynamic decision networks for decision-making in self-adaptive systems: A case study," in *Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, ser. SEAMS '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 113–122. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487336.2487355

[5] N. Bencomo and A. Belaggoun, "A world full of surprises: Bayesian theory of surprise to quantify degrees of uncertainty," in *Companion Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE Companion 2014. New York, NY, USA: ACM, 2014, pp. 460–463. [Online]. Available: http://doi.acm.org/10.1145/2591062.2591118

[6] L. Chung and J. C. S. do Prado Leite, "On non-functional requirements in software engineering," in *Conceptual modeling: Foundations and applications*. Springer, 2009, pp. 363–379.

[7] S. A. Butler, "Security attribute evaluation method: A cost-benefit approach," in *Proceedings of the 24th International Conference on Software Engineering*, ser. ICSE '02. New York, NY, USA: ACM, 2002, pp. 232–240. [Online]. Available: http://doi.acm.org/10.1145/581339.581370

[8] S. Kullback, *Information Theory and Statistics*. Wiley, 1959.

[9] R. Kazman, M. Klein, and P. Clements, "Atam: Method for architecture evaluation," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-2000-TR-004, 2000.

[10] D. Bunn, *Applied Decision Analysis*. New York: McGraw Hill, 1984.

[11] S. A. Butler and P. Fischbeck, "Multi-attribute risk assessment," in *Symposium on Requirements Engineering for Information Security*, 2002.

[12] M. Harman, W. B. Langdon, Y. Jia, D. R. White, A. Arcuri, and J. A. Clark, "The gismoe challenge: Constructing the pareto program surface using genetic programming to find better programs (keynote paper)," in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE 2012. New York, NY, USA: ACM, 2012, pp. 1–14. [Online]. Available: http://doi.acm.org/10.1145/2351676.2351678

[13] 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*, ser. SIGSOFT '04/FSE-12. New York, NY, USA: ACM, 2004, pp. 53–62. [Online]. Available: http://doi.acm.org/10.1145/1029894.1029905

[14] R. Ali, C. Solis, I. Omoronyia, M. Salehie, and B. Nuseibeh, "Social adaptation: when software gives users a voice," 2012.

[15] A. Filieri, C. Ghezzi, and G. Tamburrelli, "A formal approach to adaptive software: Continuous assurance of non-functional requirements," *Form. Asp. Comput.*, vol. 24, no. 2, pp. 163–186, Mar. 2012. [Online]. Available: http://dx.doi.org/10.1007/s00165-011-0207-2

[16] M. J. Carr, S. L. Konda, I. Monarch, F. C. Ulrich, and C. F. Walker, "Taxonomy-based risk identification," DTIC Document, Tech. Rep., 1993.

[17] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *Software, IEEE*, vol. 14, no. 5, pp. 67–74, Sep 1997.

[18] K. Deb and S. Gupta, "Understanding knee points in bicriteria problems and their implications as preferred solution principles," *Engineering optimization*, vol. 43, no. 11, pp. 1175–1204, 2011.

[19] P. Grace, D. Hughes, B. Porter, G. S. Blair, G. Coulson, and F. Taiani, "Experiences with open overlays: A middleware approach to network heterogeneity," in *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008*, ser. Eurosys '08. New York, NY, USA: ACM, 2008, pp. 123–136. [Online]. Available: http://doi.acm.org/10.1145/1352592.1352606

[20] D. Hughes, P. Greenwood, G. Coulson, and G. Blair, "Gridstix: Supporting flood prediction using embedded hardware and next generation grid middleware," in *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, ser. WOWMOM '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 621–626. [Online]. Available: http://dx.doi.org/10.1109/WOWMOM.2006.49

[21] N. Bencomo, K. Welsh, P. Sawyer, and J. Whittle, "Self-explanation in adaptive systems," in *Proceedings of the 2012 IEEE 17th International Conference on Engineering of Complex Computer Systems*, ser. ICECCS '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 157–166. [Online]. Available: http://dx.doi.org/10.1109/ICECCS.2012.34

[22] T. E. Agency, "Ribble at samlesbury," November 2014. [Online]. Available: http://apps.environment-agency.gov.uk/river-and-sea-levels/120712.aspx?stationId=5122

[23] A. van Lamsweerde, "Risk-driven engineering of requirements for dependable systems," *Engineering Dependable Software Systems*, vol. 34, p. 207, 2013.

[24] S. L. Lim, D. Quercia, and A. Finkelstein, "Stakenet: Using social networks to analyse the stakeholders of large-scale software projects," in *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ser. ICSE '10. New York, NY, USA: ACM, 2010, pp. 295–304. [Online]. Available: http://doi.acm.org/10.1145/1806799.1806844

[25] C. B. Haley, R. C. Laney, and B. Nuseibeh, "Deriving security requirements from crosscutting threat descriptions," in *Proceedings of the 3rd International Conference on Aspect-oriented Software Development*, ser. AOSD '04. New York, NY, USA: ACM, 2004, pp. 112–121. [Online]. Available: http://doi.acm.org/10.1145/976270.976285

[26] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004. [Online]. Available: http://dx.doi.org/10.1109/TDSC.2004.2

[27] T. Anderson, *Resilient Computing Systems*. John Wiley & Sons Australia, Limited, 1990. [Online]. Available: https://books.google.com.sa/books?id=nBPIkQEACAAJ

[28] J. Camara and R. de Lemos, "Evaluation of resilience in self-adaptive systems using probabilistic model-checking," in *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop on*, June 2012, pp. 53–62.

[29] E. Letier, D. Stefan, and E. T. Barr, "Uncertainty, risk, and information value in software requirements and architecture," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 883–894. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568239

[30] J. Whittle, P. Sawyer, N. Bencomo, and B. Cheng, "A language for self-adaptive system requirements," in *Service-Oriented Computing: Consequences for Engineering Requirements, 2008. SOCCER '08. International Workshop on*, Sept 2008, pp. 24–29.

[31] A. Elkhodary, N. Esfahani, and S. Malek, "Fusion: A framework for engineering self-tuning self-adaptive software systems," in *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE '10. New York, NY, USA: ACM, 2010, pp. 7–16. [Online]. Available: http://doi.acm.org/10.1145/1882291.1882296

[32] K. J. Sullivan, W. G. Griswold, Y. Cai, and B. Hallen, "The structure and value of modularity in software design," in *Proceedings of the 8th European Software Engineering Conference Held Jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. ESEC/FSE-9. New York, NY, USA: ACM, 2001, pp. 99–108. [Online]. Available: http://doi.acm.org/10.1145/503209.503224