A World Full of Surprises: Bayesian Theory of Surprise to Quantify Degrees of Uncertainty

Nelly Bencomo
Aston University, UK
nelly@acm.org

Amel Belaggoun
CEA, LIST, France
amel.belaggoun@cea.fr

ABSTRACT

In the specific area of software engineering (SE) for self-adaptive systems (SASs) there is a growing research awareness about the synergy between SE and artificial intelligence (AI). However, just few significant results have been published so far. In this paper, we propose a novel and formal Bayesian definition of *surprise* as the basis for quantitative analysis to measure degrees of uncertainty and deviations of self-adaptive systems from normal behavior. A *surprise* measures how observed data affects the models or assumptions of the world during runtime. The key idea is that a "surprising" event can be defined as one that causes a large divergence between the belief distributions prior to and posterior to the event occurring. In such a case the system may decide either to adapt accordingly or to flag that an abnormal situation is happening. In this paper, we discuss possible applications of Bayesian theory of surprise for the case of self-adaptive systems using Bayesian dynamic decision networks.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.2 [**Software Engineering**]: Design Tools and Techniques

General Terms

Theory

Keywords

self-adaptation, uncertainty, bayesian networks, bayesian surprise

1. INTRODUCTION

Surprises do not exist in a 100% deterministic and predictable world. In other words, surprises exist only in the presence of uncertainty. Uncertainty can be caused, among other reasons, by "inherent stochasticity, missing information and limited computing resources" [4]. In the case of self-adaptive systems inherent stochasticity happens due to non-deterministic changes in the environment, missing information during requirement specification and limited computing resources during runtime [7]. We believe there is a great potential in AI techniques to provide benefits to SE and specifically in SE applied to the engineering process of SASs. The definition given here includes concepts based on Bayesian Networks[8] and prior and posterior distributions to, therefore, deal with uncertainty and express and achieve expectations.

We think we are the first to propose an approach that uses Bayesian surprise (i.e. a considerable divergence between the belief distributions prior and posterior to the occurrence of an event) for the case of SASs. The main contributions of this paper are: (i) a novel and sound approach to support decision making in order to, for example, flag up that an abnormal situation is happening and that an adaptation may be required, (ii) a proof of concept that our approach can be applied to real applications

2. BAYESIAN SURPRISE: A MATHEMATICAL DEFINITION OF SURPRISE

Given a small set of axioms, Bayesian theory of probability provides a way for modeling and reasoning about uncertainty. In [4] Baldi and Itti propose a formal Bayesian definition of surprise to measure how data can affect a given observer. Essentially, the effect of (observed) data on the observer is the transformation of the observer's prior beliefs into posterior beliefs, according to Bayes' theorem. The changes are therefore considered in terms of the distance between posterior and prior beliefs about the world. Baldi and Itti based their proposed definition on the fact that the Bayesian theory allows us to associate degrees of belief or confidence with hypothesis about the world or models. Furthermore, we can denote these degrees of belief using real numbers rescaled to the [0,1] interval (i.e. referring to them as probabilities). In terms of probability theory, how much an observer is surprised can be regarded as the distance between the prior and the posterior probability distributions. Consider an observer with prior distribution P($\mathcal{M}$) over a set of $\mathcal{M}$ of possible models or assumptions.

$$\{P(M)\}, M \in \mathcal{M}$$

Suppose D specifies that some specific data was observed. The event D leads to a reevaluation of the assumptions and the possible change of the prior distribution into a posterior distribution according to Bayes' theorem:

$$P(M|D) = \frac{P(D|M)}{P(D)} P(M) \quad (1)$$

The set $\mathcal{M}$ may for instance be the following hypothesis or models of the world [4]:
$\mathcal{M}$ = { it will rain tomorrow, I have access to my bank accounts, my credit card information is secure, etc.}

For each of these hypotheses or models M a probability function exists, P(D|M), which measures the chances of D given that M is

ICSE Companion'14, May 31 – June 7, 2014, Hyderabad, India
Copyright 2014 ACM 978-1-4503-2768-8/14/05...$15.00
http://dx.doi.org/10.1145/2591062.2591118

correct. From equation (1), the observation of D changes P(M) to P(M|D). In the framework defined, the observation of D offers no surprise if it leaves the beliefs (i.e. hypothesis) of the observer unchanged, in other words if the posterior and prior distributions are the same. On the other hand, D is surprising if both distributions differ considerably. If we use the example above and the equation (1), the event D consisting of the realization that the password has been forgotten or misplaced would generate surprise. This is due to the fact that posterior beliefs in the hypothesis "*I have access to my bank account*" and "*my credit card information is secure*" would have been considerably lower than the original beliefs in the hypothesis. What is described above may result in a large divergence between the belief distributions prior and posterior to the data observation D, and therefore in big surprise. In the following sections, we introduce and discuss possible applications of Bayesian theory of surprise in the context of SASs using Bayesian dynamic decision networks.

3. DYNAMIC DECISION NETWORKS, DESIGN ASSUMPTIONS AND SASS

In [1, 2] we have proposed the use of design assumptions (i.e. assumptions made at design time) and Bayesian reasoning (using dynamic decision networks) as a means to reason about possible sources of uncertainty in dynamically adaptive systems. At runtime when the running system detects that an assumption has been falsified the system may decide to self-adapt accordingly. Design assumptions are also known as Claims which is a term used as a synonym in this paper.

In this section, we briefly offer a background about DDNs and explain in short our own approach to support decision-making under uncertainty using dynamic-decision networks (DDNs) [2] to set the needed basis to explain Bayesian surprise later in the paper.

3.1 Dynamic Decision Networks

Decision networks (DNs) [3] are graphical model representations of decision problems under uncertainty. DNs can be treated as an extension of Bayesian networks, augmented with decision and utility nodes. A decision node represents the possible alternatives the decision maker can choose. $U(X, d_j)$ in DNs are used to provide a quantitative measure of preferences among possible world states represented by the chance node X when making the decision d_j (i.e. for each value x_i of the random variable X [3]). To decide among alternative decisions, the expected utility EU_j of each alternative d_j, given evidence e, is calculated by the sum of the utilities of all possible future states of the world that follow that alternative, weighted by the probabilities of those states occurring. The expected utility is computed using the equation (1) as follows:

$$\mathbf{EU(d_j|e)} = \sum_{\mathbf{x_i} \in \mathbf{X}} \mathbf{U(x_i, d_j)} \times \mathbf{P(x_i \mid e, d_j)} \qquad (2)$$

In equation (1) above, $P(x_i \mid e, d_j)$ is the conditional probability of $X = x_i$ given the evidence e and the decision d_j. Solving a DN refers to finding the decision that maximizes EU.

Different from BNs and DNs that do not offer mechanisms for representing temporal relations between and within the random variables, DDNs are used instead. To cope with time varying nodes, DDNs maintain a series of time slices to represent nodes at consecutive periods of times. An arc that connects a node from a previous time slice to a node in the next time slice encodes an effect on the node's value by the previous node and represents what is called the transition model. Mathematically the transition model is represented by the probability $P(X_{t+1}|X_t, d_t)$. The arc that connects E_t and X_t relates to the conditional distribution $P(E_t|X_t)$. Such a probability is known as the observation model or sensor model. E_t represents the set of observable evidence variables. The observation at time t is represented by $E_t = e_t$ for some set of values e_t [8]. The observation model describes how the actual state of the world (i.e. indicated by chance nodes X) affects the evidence variables [8].

3.2 DDNs Model for SASs

In [2] we have presented an approach based on the mathematical model provided by DDNs which (i) underpins the decision-making for self-adaptation under uncertainty and (ii) is driven by levels of satisficement of non-functional requirements (NFRs). Decisions in the DDN correspond with alternative design decisions d_j that correspond with different configurations a SAS can undertake. The random variables associated with the chance nodes in the DDN represent the levels of satisficement of NFRs given the different design decisions and the corresponding configurations. The conditional probability P(NFR$_i|d_j$) represents the probability of NFR$_i$ being satisficed given a decision d_j. For each NFR$_i$, a function w(NFR$_i$,d_j) over every design alternative d_j considered is identified. The function w(NFR$_i$,d_j) allows weights (i.e. preferences) to be associated with the NFRs.

The global expected utility of a DDN provides the utility function that uses the values of the conditional probabilities, and the preferences and priorities using weights over the NFRs. Therefore, using the probability values P(NFR$_i|d_j$) and the weights w(NFR$_i$,d_j), the expected utility of d_j can be computed.

The initial conditional probabilities are either estimated by experts or compiled from previously gathered statistics [10]. These conditional probabilities will be updated by the DDN by probabilistic inference (i.e., Bayesian updating). Probabilistic inference occurs whenever new evidence arrives. As a result, DDNs can provide a quantitative technique to make informed decisions due to the arrival of new evidence during either runtime.

Crucially, changes on the conditional probability functions and its values (or beliefs) due to learned information cause the need to reevaluate the DDN. Therefore, environmental properties (i.e., properties of the operational context) that can cause changes on the probability distributions and, consequently, on the conditional probabilities of the chance nodes need to be identified accordingly. We call those environmental properties, uncertainty factors.

These uncertainty factors are monitored using monitorables (i.e., entities in the environment and the system itself that can be monitored [2]) to produce the evidence needed to trigger the decision-making process given the changes in the systems beliefs. Monitorables are sensors in the monitoring infrastructure that can observe and provide information to determine the values of the environmental properties that correspond to the uncertainty factors. The uncertainty factors are associated with non-static evidence nodes linked to NFRs (evidence nodes that are not related to evidence nodes are considered static). In our case studies, the monitorable are used to verify the validity of design assumptions or Claims.

So far, our approach has been used in different cases including two real applications: a Remote Data Mirroring [2] System and a Sensor Network for River monitoring.

4. EXPERIMENTS WITH BAYESIAN SURPRISE USING DDNS

"Surprises" can quantify the unanticipation of new data observation given the current model of the environment. To quantify the surprise factor of an observation and following the definition given in Section 2, we measure the amount of information gained by a set $\mathcal{M}$ of models. We compare the probability of a model

M given a new data observation D, P(M|D), against its prior P(M) over all models M∈ $\mathcal{M}$. The surprise can be measured using the Kullback-Leibler divergence (KL) [5], which estimates the divergence between the prior and posterior distributions.

The surprise factor is measured using the information gain as:

$$S(D,\mathcal{M}) = KL(P(M|D),P(M)) = \sum_{\mathcal{M}} P(M|D)\log\frac{P(M|D)}{P(M)} \quad (3)$$

According to (3) if the posterior distribution is the same as the prior distribution then, there is no surprise. Also, based on the expression above, a unit of surprise called *wow* can be applied. *wow* has been defined for a single model M as the amount of surprise corresponding to a two-fold variation between P(M|D) and P(M), i.e. as log P(M|D)/P(M)) with log taken in base 2 [4].

4.1 Experiments Monitoring One Claim

In [2], we report results of the application of our approach to the case study of a Remote Data Mirroring (RDM) application. A simplification of the case is as follows.

RDM is a classic technique whose goal is to protect data against inaccessibility, and to provide resistance to data loss. For example, different network topologies pose different costs and benefits and therefore different trade-offs need to be done when choosing one. A redundant topology (RT) offers a higher level of reliability than a minimum spanning tree topology (MST). However, the costs of maintaing a non-stop redundant topology may be prohibitive. An RDM application can be configured according to (i) alternative network topologies such as MST and redundant topology; or (ii) the way how data distribution is achieved, e.g., synchronous and asynchronous (not taken into account here). Each configuration provides different levels of data protection, performance and costs which are quality properties. In the study in [6], three quality properties were mapped to the following NFRs: *Minimize Operational Cost MO*, *Maximize Reliability MR* and *Maximize Network Performance MP*.

A DDN for the application RDM described has been designed to support the decision-making of choosing between the configurations MST and RT while new information is gathered during execution. Different tests have been executed. The results of tree of the experiments performed with the DDN are shown in Table 1. Fig. 1 shows the results of the computation of the expected utility (EU) for each configuration during time slices t=0 to t=8 for the experiment 1. The experiments 1,2 and 3 evaluate how evidence about the validity (or falsification) of the recorded assumption C1 ="Redundancy prevents network partition" can trigger the need of runtime adaptations for the RDM system. C1 has an effect on the probability distribution of the NFR MR making it a non-static node in the DDN (denoted by MRt). C1 is falsified if two or more network links fail simultaneously. The other two chance nodes in the DDN related to the NFRs MO and MP remain static and therefore do not feed into to surprise factors.

Based on [6], we have considered the following initial conditional probabilities P(NFR$_i|d_j$):

P(MP =*true* |MST Topology)= 0.5, P(MP =*true*|Redundant topology)= 0.5,
P(MO =*true* |MST Topology)= 0.75, P(MO =*true*|Redundant topology)= 0.25.
$P(MR_t$=*true*|MST Topology)= 0.25, $P(MR_t$=*true*|Redundant Topology)= 0.95,
$P(E_1$ =*true* |MR_t=*true*)= 0.99, $P(E_2$=*true* |MO =*true*)= 0.999

Given the initial conditional probabilities and the utilities provided by experts before runtime, the DDN suggests that the best decision is to "Use Redundant Topology" as the initial configuration. This configuration is based on the validity of the assumption C1 ="Redundancy Prevents Network Partitions" that states that a redundant network topology prevents network link failures from partitioning the network [6]. Later, at time slice 3, new information is collected that concludes the assumption C1 is not longer true and therefore EU(MST Topology)>EU(Redundant Topology) and the choice "Use MST Topology" is considered by the DDN as the best decision this time. It is the most suitable as the use of "Redundant Topology" decision does not necessarily prevent network partitions any more. At time slice 7, the monitoring infrastructure finds the claim C1 is valid again and the DDN correctly suggests to adapt to the original design decision "Use Redundant Topology". Figure 1 shows the results of this experiment 1.

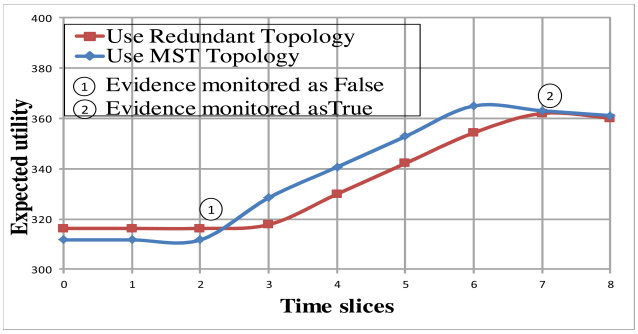

Figure 1: EUs during runtime and adaptations for Experim 1

Surprise Computation: The same scenario as described above but using different utilities and initial probabilities was run for the experiment 2. The adaptations are highlighted in the corresponding rows 3 and 7 for each experiment in Table 1 in the column Adaptation.

To calculate the surprise S we use equation (3). The probability distributions P(MRt|MST) and P(MRt|RT) are used. The model space and data to be observed are:

$\mathcal{M}$ = { Maximize Reliability MRt } and D = "C1 validity changes".

Table 1 also shows the surprise values associated with the time slices from 0 until 8 in each experiment. Note that for the case of experiments 1 and 2, each time slice where an adaptation was suggested by the DDN, effectively a "surprise" was found as expected as the KL values for slice times 3 and 7 are greater than 0.

Table 1: Summary of results of experiments

Experiments	*Time slices*	Adaptation	S
	0	–	0
	1	–	0
	2	–	0
Experim. 1	3	yes	**1.7204 wow**
	4	–	0
	5	–	0
	6	–	0
	7	yes	**7.1093 wow**
	8	–	0
	0	–	0
	1	–	0
	2	–	0
Experim. 2	3	yes	**2.095 wow**
	4	–	0
	5	–	0
	6	–	0
	7	yes	**7.6229 wow**
	8	–	0
	0	–	0
	1	–	0
	2	–	0
Experim. 3	3	yes	**1.6195 wow**
	4	–	0
	5	–	0
	6	–	0
	7	no	**6.828 wow**
	8	–	0

For the case of Experiment 3, again the same scenario as described above was applied but this time different weights (preferences) were used in time slice 7 in such a way that a big preference-value for MST was used, i.e. even if the probabilities of meeting the NFR given RT were better, the high preferences for MST made the DDN to keep MST as the better choice, and therefore different from the other 2 experiments an adaptation was not suggested by the DDN in the time slice 7. As the KL-based surprise factor does not use the preferences values but just the probability distributions, a surprise is given place even if the DDN does not suggest an adaptation. Therefore, we conclude that the Bayesian surprise can be used to flag up situations where biased preferences can hide the need to perform an adaptation, i.e. the Bayesian surprise could be used to study and perhaps correct unwanted effects of preferences.

4.2 Experiments Monitoring Two Claims

The generic scenario that has been used to perform the new experiments is as follows: the two possible architectural choices are the same as before "Use MST Topology" and "Use Redundant Topology". The states of two design assumptions "C1 = Redundancy prevents the networks partitions" and "C2 = Fewer active network links reduce operational costs" are monitored during runtime. The observed values C1 and C2 are shown in Table 2. C1 has an effect on the probability distribution of the NFR MR making it a non-static node in the DDN (denoted by MRt). C2 has an effect on the probability distribution of the NFR MO making it a non-static node (denoted by MOt). The NFR MP remains static.

Table 2: Monitored values of C1 and C2

slice time	C1	C2
0	T	T
1	F	T
2	F	T
3	T	F
4	T	F
5	T	T
6	F	T
7	T	T

Surprise Computation: The model spaces and data to be observed are

$\mathcal{M}1$={Maximizing Reliability (MRt)}, D1="C1 validity changes".

$\mathcal{M}2$={Minimizing Oper. Costs (MOt)}, D2="C2 validity changes".

Two surprises S1 and S2 are calculated using equation (3). The probability distributions P(MRt|MST), P(MRt|RT) and P(MOt|MST), P(MOt|RT) are used to calculate S1 and S2. The prior models for surprise computation are P(MRt) and P(MOt) and the posterior models when D1 and D2 have been observed are P(MR|C1) and P(MO|C2) respectively. Table 3 shows the computation of the surprises S1 and S2 based on the KL-divergence between the prior and the posterior probabilities. As above, surprises take place in the same time slices (1,3,6,7) where the DDN has suggested adaptations. However, in the slice time 5 a surprise is not suggested by S1 even if it is suggested by S2. Examining the weight values given to the RT choice (while meeting the NFRs MR and MO) and MST choice (while meeting the NFRs MR and MO), we saw that the preferences for RT were almost the double of the preferences for MST. This choice, certainly, may not be a good selection for the current situation as using RT would create unnecessary costs as NFRs would be met using the less costly topology MST. The surprise S flags up this situation.

Table 3: Summary of experiments

Time slices	Adaptation	S1	S2
0	–	0.0	0.0
1	yes	0.634040 wow	0.732208 wow
2	–	0.0	0.0
3	yes	0.017348 wow	5.702488 wow
4	–	0.0	0.0
5	–	0.0	9.327813 wow
6	yes	10.273983 wow	0.029776 wow
7	yes	6.063689 wow	0.044626 wow

5. CONCLUSIONS

The results achieved so far are encouraging and open different research avenues that certainly require expertise from both research areas AI and SE used for self-adaptation. Machine learning techniques offer a promising path to be able to tackle uncertainty that need to be studied further together with SE techniques.

The presented results even if preliminary are novel and promising. Bayesian surprise could be used to explore the operating environment to therefore improve its understanding. With further broader experiments, we plan to: (i) search further meanings of the surprise values. For example, we want to further interpret the size of a surprise (i.e. other uses of the wow unit). One of the benefits of being able to characterize if a surprise is small is that the surprises could be used as a way of providing an implementation of the RELAX language [9]. Furthermore, small surprises could be used to tolerate evidence of unanticipated but transient environmental conditions that can trigger unnecessary adaptations. Doing so would be an alternative solution to the one offered in [6]. (ii) to further study how Bayesian surprise can support sensitivity analysis (to agree on consistent utility functions) to support decision-making provided by the DDNs. (iii) study alternative ways to measure a surprise that do not use KL (in order to make further comparisons).

6. ACKNOWLEDGMENTS

We thank Prof. Jon Whittle for the discussions on the early ideas about Bayesian Surprise.

7. REFERENCES

[1] N. Bencomo and A. Belaggoun. Supporting decision-making for self-adaptive systems: From goal models to dynamic decision networks. In *REFSQ*, pages 221–236, 2013.

[2] N. Bencomo, A. Belaggoun, and V. Issarny. Dynamic decision networks for decision-making in self-adaptive systems : a case study. SEAMS 2013, 2013.

[3] R. Howard and J. Matheson. Influence diagrams. In *Readings on the Principles and Readings on the Principles and Applications of Decision Analysis II*, Menlo Park CA:, 1984.

[4] L. Itti and P. F. Baldi. Bayesian surprise attracts human attention. *Vision Research*, 49(10):1295–1306, May 2009.

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

[6] A. Ramirez, B. Cheng, N. Bencomo, and P. Sawyer. Relaxing claims: Coping with uncertainty while evaluating assumptions at run time. *MODELS*, 2012.

[7] A. Ramirez, A. Jensen, and B. Cheng. A taxonomy of uncertainty for dynamically adaptive systems. In *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012*, pages 99 –108, june 2012.

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

[9] J. Whittle, P. Sawyer, N. Bencomo, B. Cheng, and J.-M. Bruel. Relax: a language to address uncertainty in self-adaptive systems requirement. *RE Journal*, 2010.

[10] H. Ziv, D. J. Richardson, and R. Klosch. The uncertainty principle in software engineering, 1996.