Enhancing Context Specifications for Dependable Adaptive Systems: A Data Mining Approach

Arthur Rodrigues[1], Genaína Nunes Rodrigues[1⋆], Alessia Knauss[2], Raian Ali[3], and Hugo Andrade[2]

[1] Universidade de Brasília, Brazil
arthy.rf@gmail.com; genaina@cic.unb.br
[2] Chalmers | University of Gothenburg, SE
alessia.knauss, sica@chalmers.se
[3] Bournemouth University, UK
rali@bournemouth.ac.uk

Abstract.

Context: Adaptive systems are expected to cater for various operational contexts by having multiple strategies in achieving their objectives and the logic for matching strategies to an actual context. The prediction of relevant contexts at design time is paramount for dependability. With the current trend on using data mining to support the requirements engineering process, this task of understanding context for adaptive system at design time can benefit from such techniques as well.

Objective: The objective is to provide a method to refine the specification of contextual variables and their relation to strategies for dependability. This refinement shall detect dependencies between such variables, priorities in monitoring them, and decide on their relevance in choosing the right strategy in a decision tree.

Method: Our requirements-driven approach adopts the contextual goal modelling structure in addition to the operationalization values of sensed information to map contexts to the system's behaviour. We propose a design time analysis process using a subset of data mining algorithms to extract a list of relevant contexts and their related variables, tasks, and/or goals.

Results: We experimentally evaluated our proposal on a Body Sensor Network system (BSN), simulating 12 resources that could lead to a variability space of 4096 possible context conditions. Our approach was able to elicit subtle contexts that would significantly affect the service provided to assisted patients and relations between contexts, assisting the decision on their need, and priority in monitoring.

⋆ Corresponding author - Postal Address: Dept. de Ciência da Computação - Campus Darcy Ribeiro, Edifício CIC/EST, Asa Norte, Brasília - DF - Brasil, Postal Code: 70910-900.

Conclusion: The use of some data mining techniques can mitigate the lack of precise definition of contexts and their relation to system strategies for dependability. Our method is practical and supportive to traditional requirements specification methods, which typically require intense human intervention.

Keywords: Self-adaptive system, Context uncertainty, Data mining, Design time, Goal modelling, Dependability

1 Introduction

Adaptivity to the various contexts in which the system operates is a key feature of a (self-)adaptive system (SAS). Analysts seek to know as many contextual factors as possible, i.e., elements that reify the environment in which the system operates, and seek to provide strategies in the system that fits their various values. The literature in the area of context elicitation shows the high effort needed to properly determine the conditions for system adaptation at early stages of software development [1,2,3]. A key difficulty in determining such conditions early on is the lack of information at initial stages of development which limits the soundness and completeness of context specifications in terms of identifying relevant contextual variables, their combination, and their effect on the suitability and quality of a certain system strategy to achieve a system goal. Such specification can be subjective by nature and there could be different views among stakeholders that often lead to a system with requirements that are neither alternative nor consistent and there could be a disagreement for the specification on how they are triggered and enabled by specific contextual conditions [2].

Context shall be seen as an integral part of the specification of requirements, especially in the case of SAS, which are usually non-uniform and context-dependent [4]. Taking context into consideration at the requirements level is essential for a better design of SAS. It allows early decision making and provides rationales on design decision by filtering the applicable strategies and enabling the detection of possible conflicts between them in certain contexts [2]. Missing important contextual factors can lead to decisions which are incorrect. Similar to experimental research, some variables can prove to be confounding or moderating. The identification of variables that could be of relevance is itself a complex process.

In the realm of SAS, dependability attributes such as reliability, availability, safety, security, and maintainability are important design drivers [5]. The correct specification of (contextual) requirements for such systems is therefore essential. Knowing in advance the contexts which a SAS will be exposed to is fundamental to define before the actual operation of the system. This will enable estimating the dependability level and providing corrective measures early on. The dependability of a self-adaptive system can also be affected by the uncertainty of possible operational contexts [6,7], that is, any unexpected state of the world that hinders or prevents the system from fulfilling the requirements. This may not be only due to limitations in the elicitation stage but also due to possible faults in the monitoring infrastructure of such contexts. The availability of the resources needed to accomplish the system tasks, including the monitoring process, is a dynamic property by itself [8]. At design time, our method focuses on avoiding sources of uncertainties arising from the noise in sensing and execution context by classifying monitoring mechanisms that might not be able to accurately perceive

contextual requirements and their evolution. We use data mining to identify inconsistencies in the composition and integration of data that comes from different sources of information. Finally, the classification and prediction routines used by the method reduce the future uncertainty of information that are relevant for decision making [7]. Our method focuses on systems which have contexts categorized as *known knowns* and *known unknowns* as per Rumsfeld's taxonomy applied to requirements [9]. One possible perspective on tackling unanticipated changes is to integrate adaptation (machine-driven process to deal with *known unknowns*) with evolution (human-driven process to deal with *unknown unknowns*) [10]. Our method brings a new perspective, which is based on supporting the requirements elicitation process to reduce known unknowns. This helps determining and understanding the context.

Despite the recognition of the benefits of a thorough requirements engineering process, especially on challenging issues such as *unknown unknown* requirements [11], only a few attempts considered the elicitation and refinements of contextual requirements [12,13,2], while the majority of work focused on conceptual modelling and automated reasoning [14,15,7,16]. This is particularly true for context elicitation for dependability. There is a lack of research considering the impact that contexts have on dependability and other quality attributes. By using the term contextual requirements engineering, we focus on the elicitation and refinement of the contexts that may have a strong influence on users goals, on the way to reach them, and on the quality of each way [4]. Through such a process, it is noticeable that the system can adapt to the context of its stakeholders and offer the applicable requirement, which is paramount to the development of adaptive systems with a large number of end-users [2]. While existing techniques on context elicitation assume manual identification with user input, this may not scale for the size of contextual factors. As the system dynamics change, existing knowledge tends to become invalid, and needs to be refined. Moreover, contextual factors may hinder the elicitation of requirements itself [17]. It is unfeasible to make a combinatorial exploration of every monitored attribute and map of all possible contexts afterwards. Such approach will result in a state explosion problem. Runtime analysis can also be risky as faults may not be tolerated for dependable adaptive systems. The use of prototypes helps to overcome this limitation, and, at the same time, supports the understanding of the context from different end-user perspectives [2]. It can potentially help to reduce failures caused by invalid adaptations. The use of runtime context data, generated through simulation or past system experiences, can be a promising alternative to automatically detecting a range of possible context conditions at design or maintenance time.

In this paper, we propose a data mining supported approach to optimize and refine the specification of context during the requirements elicitation phase. The approach is able to identify contextual variables and combinations of such variables, mitigating uncertainty at early stages and providing evidence to decide which strategy the system should adopt in order to fulfil the requirements. It also aims at identifying priorities in the monitoring process as some contextual variables are prior to others in the decision making process. Since goal modelling is a main concept for self-adaptive systems on how to depict and formalize the knowledge around requirements and context of such systems [18], our approach is goal-driven and uses a Contextual Goal Model (CGM) [4] to map the contexts to the system's alternative strategies to achieve requirements. Our

approach requires a complementary qualitative context analysis activity to update the contextual constraints into their respective nodes of the model.

We evaluate our approach on a simulated prototype version of a Body Sensor Network system (BSN) [19]. The results showed the ability of the approach to detect important and critical cases to take corrective actions. It is also capable of detecting additional rules to apply to optimize monitoring context.

The remaining parts of this paper are organized as follows: In Section 2, we provide a succinct characterization of the Body Sensor Network system, a brief introduction to contexts, contextual modelling, and some background on data mining. In Section 3, we present the core of our proposal, followed by Section 4, where we report the evaluation of our approach. Section 5 elaborates on major related work. Finally, Section 6 concludes the paper along with future work.

2 Background

2.1 Running Example: Body Sensor Network

Over the past decades, the industries' capacity to miniaturise and reduce costs of semiconductors and other electronic components enabled the development of tiny and affordable computers, with a processing power that grows every year. In a similar way, improvements in wireless communication, sensor design, and energy storage technologies allowed the creation of truly pervasive Wireless Sensor Network [20]. Merging these ideas with research in the medical domain makes it possible to create integrated devices for body parameters monitoring so called Body Sensor Networks (BSN). We illustrate the concepts of our approach throughout this work using the example of the Body Sensor Network proposed in [19,21]. As the cornerstone of this work is the improvement of context specification for dependability analysis, we have chosen the BSN example as for this system the dependability attributes are of highest importance due to health issues. Such a system help us to explore and understand the impact of contexts on dependability attributes better, such as reliability and availability. Moreover, we have chosen a running example that has self-adaptive traits (e.g., self-configuration and self-management) or at least is easily upgradeable to a SAS. Finally, the BSN is prone to the effects of context variability, especially due to the variety of end users. The BSN is not the most complex adaptive system, but even for this example our approach was able to discover many different contexts which an analyst would struggle to precisely define. This could be an indicative that complex systems can benefit even more from our method

The BSN is composed of wireless sensors that are connected to a person. There may be a central node (Control Sensor) responsible for preprocessing the data collected, filtering redundancy, or translating communication protocols. The other sensors are the following: Accelerometer (Acc.), ECG (for heart rate and electrocardiogram curve), Oximeter (for blood oxidation and blood oxidation curve, called SPO2), and Temperature (Temp).

The main objective of a BSN system is to continuously monitor the vital signs of an individual adjusting its configuration according to the patient's health risk status. Thus, the prevalent self-* property that makes the BSN a typical self-adaptive system is

the self-configuration. The adjustment lies in turning on/off some sensors or changing their sampling rate. The individual's health risk can be classified into *low*, *normal*, or *high risk* categories. The patient's risk status maps to a quality goal of BSN. Each health risk requires a minimum quality level to be considered trustworthy, while each different configuration of BSN provides a different level of reliability to the operation. However, one or more resources might be unavailable at certain moments depending on the reliability of each device.

Patients with different profiles should be able to wear the system, and, over time, these profiles or even the subsidiary medical knowledge may evolve. Hence the quality goals need to adjust according to each profile. The ranges of the sensors involved in the operationalization of the patient status was defined by a health expert and declaratively expressed in the aforementioned work [19]. Table 1 shows how the sensors' values relate to the context *healthRisk*. If at least one of the resources' measurements is at high risk, the patient's health status could be identified as *"the user is at high risk"*. If the previous state does not happen and, at least one of the resources' measurements is at normal risk, the patient health status could be identified as *"the user is at normal risk"*. At last, if none of the aforementioned states is present, the patient health status could be identified as *"the user is at low risk"*.

Sensor Information	**Sensor Information Ranges**
Oxygenation:	$100 >$ low $> 94 >$ normal $> 90 >$ high > 0
Pulse Rate:	high $> 120 >$ low $> 80 >$ high > 0
Temperature:	$50 >$ high $> 38 >$ normal $> 37 >$ low $> 35 >$ normal $> 30 >$ high > 0
Fall:	if (Fall = 'yes') $\longrightarrow$ Patient's status = high risk

Table 1: Overview of the contextual variables/sensors and the classification into risk profiles for each of the contextual variables

2.2 Contexts in Self-Adaptive Systems

The word *context* carries an important and pervasive concept, not restricted to the computational field. This fact sometimes misleads the communication of the meaning of *context*, since it is common that people tacitly understand its definition but apply it in different ways in their research. For the purpose of our research, among all the well-known context definitions found in literature [22], we use the one proposed in [4], where the authors state:

> "A context is a partial state of the world that is relevant to an actor's goals." (Ali et al., 2010, p. 4).

Following such definition, contexts are monitorable pieces of information about the environment in which a system operates. Finkelstein and Savigni [23] define environment as "whatever over which we have no control", e.g. environmental conditions, user characteristics or availability of resources. The context analysis presented in [4] relies on

the refinement of contexts described at a high level of abstraction (*world predicates*) to a formula of observable facts. We interpret world predicates as descriptions, in natural language, of partial states of the world. However, systems usually do not work in such a high level of abstraction, therefore, we present the concept of *contextual variables* to represent the world predicates in a way that is understandable by a computational entity. Following this idea, we also specify a context as a formula of world predicates, and a world predicate as a formula of context variables, as presented in the Listing 1.1:

Listing 1.1: Grammar for world predicate formulas

```
Formula := World Predicate | (Formula) | Formula AND Formula | Formula OR Formula
```

According to Ali et al. [4] world predicates are categorized, based on their verifiability, into two types: *facts* and *statements*. A world predicate F is a fact for an actor A if, and only if, F can be verified by A. On the other hand, a world predicate S is a statement for an actor A if, and only if, S can not be verified by A. Such methodology is suitable for adaptive and self-adaptive systems, as their adaptation is mostly defined by verifiable facts. Different contextual information can be monitored through means of sensors, and a context may be represented by the combination of world predicates, which are a high level abstraction of a single contextual variable, or a combination of them. Henceforth in this work, a context is semantically represented as a combination of world predicates. The process of analysing the monitored environment and defining contexts in terms of variables and their values is called *context operationalization* [24]. For instance, the context *"weather"* is composed of the world predicates (*wp*) *"it is raining"* and *"it is cold"*. The *wp "it is raining"* in its turn, is represented by the combination of the contextual variables *humidity_level* $\geq$ *90* and *precipitation* == *TRUE*, while the *wp "it is cold"* is expressed by the contextual variable *temperature* $\leq$ *15*. We take this routine as the operationalization of the context *"weather"*.

The definitions above are especially adequate for our approach, since the enumeration of contexts is an important step of the process. Our approach assumes as world predicates, facts that are often taken as failures, but are actually states of an entity (e.g. user, environment, system). Such states have a certain probability to happen and are relevant to the interaction between the user and the application. By *state*, we mean any information that describes an entity's behaviour at a certain moment. An entity could be the operational environment, a user or the system itself. Our approach is adjustable to such specific contexts, respecting the overall concept described in the definitions above.

2.3 Contextual Goal Models

In order to build a proper self-adaptive system blueprint, it is required to take into consideration both, the requirements and means to achieve them, but also the contextual information that may be related to the system's operation. A contextual goal model (CGM) represents the requirements as goals, ways to meet requirements as tasks, and factors that can affect the quality and behaviour of a system as contexts [4]. The CGM presented in Figure 1 depicts the goals to be achieved by the Body Sensor Network, which is meant to detect patient emergencies based on its monitored data [19,25]. The root goal is "G1: Detect Emergency", which is performed by the actor Body Sensor Network. The root goal is divided into two subgoals: "G2: Patient status is monitored"

and “G3: Sampling rate is adjusted”. G2 is refined to “G4: Vital signs are processed”, and further, G4 is divided into other two subgoals: “G5: Vital signs are monitored” and “G6: vital signs are analysed”. Such goals are then further decomposed, within the boundary of the BSN actor, to finally reach executable tasks. The only context present in the first version of the CGM is the aforementioned *“healthRisk”*, represented by the *hR* label and mapped to subtree of goal G3. The context *hR* may assume one out of three possible world predicates. Following the grammar proposed by Ali et al., depicted in Listing 1.1, the context *hR* can be described by the logic formula: *hR* = *wp1* **or** *wp2* **or** *wp3*, where *wp1* is the world predicate *“the user is at low risk”*, *wp2* is *“the user is at normal risk”*, and *wp3* represents the world predicate *“the user is at high risk”*. The execution of the task T3 will depend on the value of the current context, that is, the higher the patient’s risk, the lower the sampling interval must be.

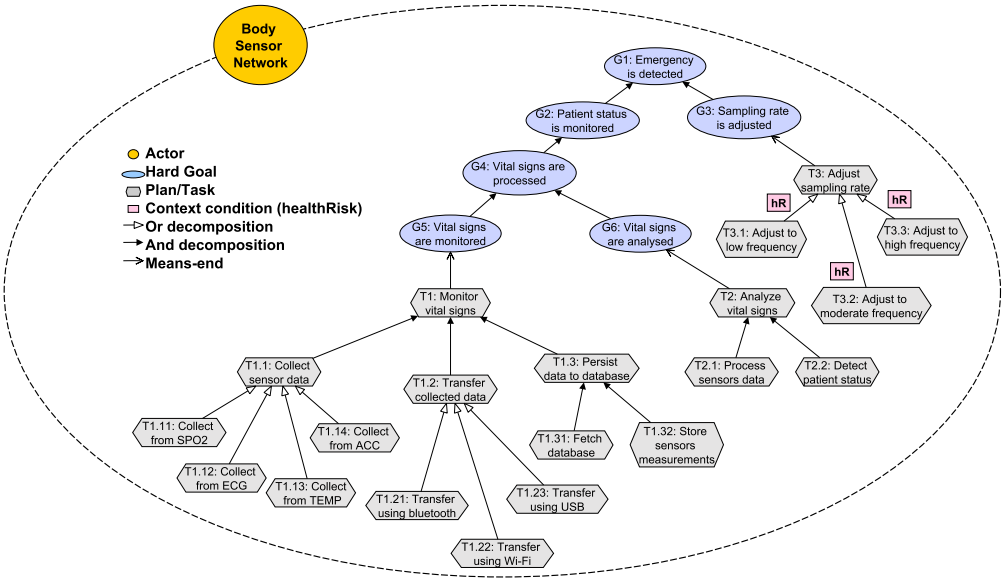

Fig. 1: CGM for the BSN system

2.4 Data Mining

With the evolution of IT ecosystems and introduction of adaptivity, traditional elicitation techniques such as interviews and ethnography have become rather time consuming and incomplete [2,26]. On the other hand, formal methods that explore every combination of environment variables searching for possible contexts are not scalable and often face problems of state space explosion (lack of memory). We claim that the combination of artificial intelligence techniques (e.g., classification algorithms) over monitored data with a robust modelling process at design time leads to a context elicitation method

that is both time and space efficient. The core of our approach relies on data mining for discovery and analysis of contextual variables and relations that can possibly lead the system to an inconsistent state. The objective of the data mining process, at design time, can be achieved by analysing historical data, for example a domain driven simulated dataset, or a prototype version, in case there is no runtime data available.

The science of machine learning is present in the fields of statistics, data mining, and artificial intelligence, intersecting with areas of engineering and other disciplines [27,28]. In self-adaptive systems, this kind of technique composes the core of the application, making possible for the system to adapt to new environments by learning from historical data [29]. In a typical operation of a self-adaptive system, we have an outcome measurement, usually quantitative or categorical, that we wish to predict based on a set of resources. We have a training set of data, coming from sensors, in which we observe the outcome and resources measurements for a set of objects (such as users of a medical monitoring service). We build a prediction model using this data which supports the prediction of new unseen objects, for instance, a new patient profile, or a future value of a given vital sign. One possible way to rate the quality of a model is through the accuracy in predicting an outcome [28].

Several techniques may be used in a data mining process [30]. Among the useful classes of data mining algorithms, there is the *Association Rule*, which uses a rule-based mechanism that allows us to discover relations between variables in large databases. The *Apriori* algorithm [31] is an example of this kind of approach. Another type of algorithm that is worth mentioning is the *Classifier*, belonging to the supervised learning class [28]. Techniques such as *JRip* (that implements a RIPPER algorithm) [32], create rules for every class in the training set and then prune these rules. The discovered knowledge in this class of algorithm is represented in the form of IF-THEN prediction rules. They are especially useful to define the operation thresholds of some resources. Lastly, we should also note the *J48* classifier algorithm, which implements a *Decision Tree*, a tree-like graph used to support the decision making process using the depth-first strategy.

While the aforementioned algorithms are often applied to provide knowledge and support the decision making in macro scenarios, like organizational and architectural levels, the novelty of our approach lies in bringing these techniques to operational level in order to identify and refine contexts that are hardly noticed by traditional elicitation techniques. If we assume that, for example, the world predicate “the patient is having a heart attack” is part of the context “emergency”, it is possible to use Apriori to discover which resources were involved in the activation of such context, for instance, heart beat sensor and accelerometer. Later on the process, JRip is used to find rules that describe the activation of the context in terms of contextual variables and their respective values. To illustrate, one could discover through a pruned rule that an individual’s pulse above 140 beats per minute indicates a risk of activating the “emergency” context. Lastly, all contextual variables can be mapped into a decision tree structure, giving a global perspective of the system to the analyst, showing how such entities interact to successfully achieve the system goals. In the next sections, we will specify the entire process and demonstrate how such combinations can take place in a sound manner.

3 Approach: Unveiling contextual variables and their combinations at design time for dependability

Our method aims at assisting the development process of dependable self-adaptive systems, by reducing as much as possible, at design time, the effects of unknown context variability. Figure 2 depicts the process. Based on the CGM structure in addition to the operationalization values of sensed information, we propose an analysis process using an adequate subset of data mining algorithms to extract a list of relevant contexts and their related variables, tasks, and/or goals. We create an execution dataset from a domain based simulation to be able to apply data mining on and identify contexts variables that are relevant. Such operational information can be provided by: (i) external sources, e.g. execution logs, test cases, data sheets of similar systems; (ii) internal sources, e.g. data generated by a simulation; or (iii) prototype versions. As an outcome of the first stage of the data mining process, the contexts as well as their operationalization are unveiled. Then the data mining process advances to the next stage and provides a quantitative impact of the contextual variables interaction into the goal achievability, given some quality constraints. A context analysis is then made in order to update the CGM by inserting the contextual constraints into their respective nodes, i.e., the tasks that have their operation modified by the specific context variability.

In the next sections, we provide a more comprehensive explanation of the steps concerning our approach.

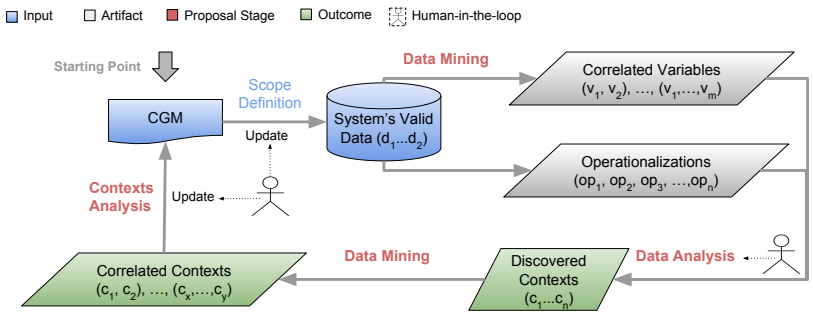

Fig. 2: Process overview of our method

3.1 Contextual Requirements in Goal Modelling

In our work, we consider that a system behaviour is fulfilled by means of the goals tree structure of the CGM. Each goal may be further refined by child subgoals, which are ultimately realized by leaf-tasks. As such, each leaf-task of the CGM dynamically realizes its following parent goals following a finite state machine previously defined in [33]. The concept of context dependency and how it is related to the system resources and their variables is encompassed by our approach. We propose an extension of the aforementioned state machine for such a purpose.

While some contexts in the CGM are directly related to the fulfilment of goals, other contexts focus on meeting quality constraints [34]. We have adapted a finite state machine from GODA [33], a goal-oriented dependability analysis framework, to represent the execution of a CGM's task in face of different kinds of contexts, as shown in Figure 3. Our proposal introduces the path of the state machine (highlighted in red), while the other paths have already been covered in [33]. From Figure 3 it can be noticed that, if there is a context that should be considered, the anticipation of it is paramount to the fulfilment of a task and therefore to its parent goals. An unsatisfied context does not only hinder the execution of a task, but might also cause a cascading error effect, compromising the fulfilment of the overall system goal. In our work, we do not only analyse the context implications from the local perspective of a task (subtree), but also from the global perspective of the tasks and their interactions towards the system's goal fulfilment.

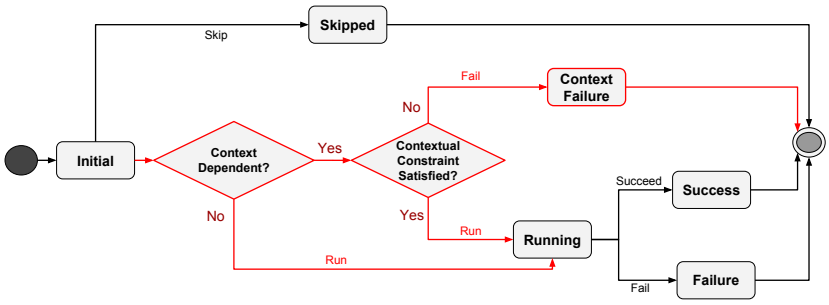

Fig. 3: Overall behaviour of CGM leaf-tasks (adapted from Mendonça et al. [33])

Initially, the self-adaptive system may try to execute or simply jump to the next task, changing its state to *Skipped*, following the task's behaviour rule. If the self-adaptive system attempts to execute a task, it is verified whether the task operation depends on any kind of context configuration. If the task is independent of context, then it proceeds to the *Running* state. Otherwise, we check whether the context is satisfied in order to proceed to *Running* or *Context Failure*. Even if the system satisfies the context constraints and proceeds to a running state, it still has a minor probability of failure (*Failure* state). This probability of failure is related to cyber-physical limitations and operational uncertainty, for example, the noise in sensing of a routing device caused by a layer of dust in its antenna. However, if the self-adaptive system does not present any operational issue, the task executes successfully.

If the task is not skipped and fails to satisfy its context conditions, we say that a *context failure* happened. Adapting Avizienis et al. domain failure classification [5], a context failure happens when the task is not able to execute under the influence of the current context (i.e., the system is not able to access the context that was defined in the contextual goal model). A context failure may characterize a *time domain*, *data domain* or both. A time domain failure happens in our approach when, for example, a resource is unavailable at the time it is requested. A data domain failure happens when the re-

quested data is out of the specified range of operation, characterizing, for example, a data integrity issue. We consider that safety may be also another dependability attribute related to both domain failure types for the context, since a failure in the monitoring infrastructure may characterize a failure in the environment [5], which could render catastrophic consequences. Table 2 summarizes the levels of domain failure and their related dependability attributes.

Domain	**Attribute**	**Consequence**
Time	Availability Safety	If a sensor is not available then the patient's data cannot be collected
Data	Integrity Safety	If a monitored data is out of range then the patient status cannot be defined

Table 2: Domain failure classification and related dependability attributes

In real case scenarios, the achievement of the main goal of a system is useless if the quality constraint is not satisfied. The definition of quality requirements we apply follows the lines of quality constraint definition from Jureta et al. [35], where "a quality constraint requires a well-defined quality space (e.g., possible values are known, values and their relationships are precisely defined, and so on) and the quality space must be shared among the stakeholders". Our method accepts that the verifiability of quality constraints requires well-defined quality spaces (e.g., as in metrics) and we need quality spaces shared among stakeholders. As such, we advocate that such quality requirements must provide a particular level of measurement resulting from our contextual variable grammar rules, as listed in 1.2. Therefore, we do not restrict quality constraints to execution components. In fact, we believe that any quantifiable non-functional requirement may be target of our analysis via measures over the execution (e.g. performance, cost, uptime, etc.). In other words, we narrowed down the expressiveness of quality constraints in our work and consider those which are possible to quantify and verify so that we enable the decision making.

At any given moment, a context is composed of one out of multiple possible world predicate combinations. The values of the contextual variables that define each world predicate may also vary over time. This core concept in our work is called *context variability*. Based on the context failure classification adopted, our method is able to deal with context variability by discovering new patterns and adjusting the valid ranges required for the operationalization of the contexts. Moreover, our method also acknowledges the concept of *context information unavailability*, that indicates a case in which the system is incapable of identifying the current context value due to a lack of information provided by the monitoring resources. Hence, context information that falls into this case is labeled as *unavailable*. The distinction between context variability and context information unavailability is made by the data analyst, with the occurrence rate of the pattern taken as an indicative parameter.

As mentioned before, a world predicate is a statement or a fact that, expressed in natural language, describes a given aspect of the world that is relevant to the decision making process of an actor. However, in a computational level, a SAS identifies a world

predicate, and by consequence a context, as a variable or a set of variables that carry the information about the environment. In our method, contextual variables are divided into different categories following the grammar defined in Listing 1.2. The types of contextual variables depend on the resources involved in their operationalization and how they interact to do so. The variables can be classified into: CV_{range}, CV_{fuzzy}, and $CV_{boolean}$ classes. The operation submitted to a CV_{range} constraint expects a numeric value (integer or float) between the range specified in the task or goal to be considered able to proceed to the running state. For instance, the world predicate "the oxygenation data is within a valid range" is represented by the following constraint over a contextual variable: Oxygenation = $\{oxyg \in \Re \mid 0 \leq oxyg \leq 100\}$). There is also a possibility of an upper or lower bound range (e.g. the world predicate "the battery level is ok" is described as the following constraint over a contextual variable: *Battery level* ≥ 15). A task or goal under a CV_{fuzzy} constraint expects a specific enumeration that represents the contextual variable in a given moment (e.g. *Patient State* == "Low Risk" **or** "Normal Risk" **or** "High Risk"). Details of the fuzzy context approach can be found in [16]. Finally, the $CV_{boolean}$ class is a contextual variable constraint in which the variable can assume boolean values only (e.g. the world predicate "the USB connection is available" is described in terms of contextual variables as: *USB* = "TRUE" **or** "FALSE"). Listing 1.2 specifies the contextual variables grammar addressing the types adopted in our method.

Listing 1.2: Contextual variable grammar and syntax rules

```
<Contextual Variable> ::= <Id> | <Expr>
<Expr>  ::= <Expr> <op_v> <Value> | <Expr> <op_r> <Range> | <Expr> <op_e> <Expr>
          |'(' Expr ')'
<op_r> ::= '<' | '<=' | '>' | '>='
<op_v> ::= '=' | '!='
<op_e> ::= '&' | '|'
<Value> ::= <Range> | <Fuzzy> | <Bool>
<Range> ::= <Int> | <Float>
<Fuzzy> ::= <Var>
<Bool> ::= ('TRUE'|'FALSE')
<Var> ::= ('a'..'z'|'A'..'Z')
<Int> ::= <Digit>|<Int>
<Float> ::= <Int>'.'<Int>
<Id> ::= <Var><Int> | <'!'><Var><Int>
<Digit> ::= '0'..'9'
```

In order to adjust the domain information to our application, we categorize the data into a resource type table describing the variables present in the CGM in accordance with our grammar, described in Listing 1.2. Table 3 presents an excerpt of a type table and some illustrative contextual variables applicable to the BSN domain. The column *Variable* represents the measurable aspects of a given resource that is involved somehow in the accomplishment of tasks and goals, i.e., the contextual variable. The *Type* column in Table 3 refers to the computational abstraction and classification of an environmental resource, i.e., aforementioned contextual variable classification. For instance, the availability of some devices (e.g. SPO2, ECG, TEMP, ACC, WiFi) are represented as a boolean data type. Meanwhile, to describe the individuals' vital signs (e.g. Temper-

ature, Pulse Rate, Oxygenation), a valid numerical range is defined following context operationalization of the BSN previously presented in Table 1. There is also the fuzzy data type, which may represent, for instance, the strength of the WiFi signal or the patient health status in a higher level of abstraction.

Contextual Variable	**Type**	**Value**	**World Predicate**
SPO2	Boolean	TRUE	The sensor SPO2 is available
Temperature	Range	$0 \leq T \leq 50$	The user's temperature is within the valid data range
Wi-Fi Signal	Fuzzy	Strong	The Wi-Fi signal is strong
Patient Status	Fuzzy	Low Risk	The user is at low risk

Table 3: Example of a resource table and description of its variables

The negation of world predicate or contextual variable can be represented by the character '!', placed before the corresponding id. However, for each class there is a specific interpretation. For the boolean class, the reasoning is straightforward, i.e., $CV_{boolean}$ means TRUE, while $!CV_{boolean}$ means FALSE. The interpretation is the same, for instance, while $CV_{boolean}$ might indicate the availability of a given resource, $!CV_{boolean}$ indicates its unavailability. The process is analogous for the CV_{range} class. Even though the context constraint consists in the expectation of a numerical value belonging or not to an specific range (e.g. $0 \leq oxyg \leq 100$), the practical implication converges to a binary decision, i.e., C_{range} represents the range constraint being respected (e.g. *oxyg* = 98), while $!CV_{range}$ represents the violation of such constraint (e.g. *oxyg* = 103). Lastly, a CV_{fuzzy} class can assume one out of multiple values (e.g. *Bluetooth_strength* = "weak" or "normal" or "strong"). The failure of such context, i.e., $!C_{fuzzy}$, represents the absence of one of the predefined contextual variables possibilities, implying the unavailability of the resource or variable analysed.

3.2 Data Mining Process

In this section, we provide a coarse-grained stepwise perspective of our context mining process through Algorithm 1. We also describe the process concretely via the execution of the algorithm on the persistence module of our running example. Design and context elicitation are creative processes, this is why our method is semi-automated and requires a human-in-the-loop step. Our algorithm does not intend to communicate that it is able to deterministically identify new contextual requirements for a software system. We already stated that our objective is to provide knowledge to support the refinement and completeness of the context specification.

Pre-processing data is an important step of most data-driven routines. We abstracted this step away from our work flow as we are mainly concerned with a conceptual model of the self-adaptive system operation. Algorithm 1 is presented to simplify the understanding of our method through a simple and tidy set of instructions.

Starting the algorithm, it receives a CGM as an input parameter. In addition, the resources syntax table, such as the illustrated Table 3, is also taken as input parameter, and contains a list of the system's resources with the respective variables name, data

Algorithm 1 ContextsMining

```
Input: ResourcesSyntaxTable resources_table, CGM cgm
Output: ContextList, ProbabilityReport
 1: CorrelatedVariables varlist ← NULL
 2: OperationalizationRules operlist ← NULL
 3: Dataset simdata ← resources_ranges.getDataset()
 4: for all n in cgm do
 5:     varlist.push(simdata.associationRule(n))
 6:     simdata ← simdata.filter(varlist)
 7:     operlist.push(simdata.classify(n), n.id)
 8: end for
 9: ContextList contexts←processRules(operlist)
10: for all records in simdata do
11:     records.replace(contexts)
12: end for
13: for all n in cgm do
14:     if contexts.getContextId(n.id) != NULL then
15:         cgm ← cgm.insert(contexts.getContextId(n.id))
16:     else
17:         return
18:     end if
19:     ContextList context_relations ← simdata.classify()
20:     ProbabilityReport probability_list ← simdata.getQuantitativeAnalysis(context_relations)
21: end for
```

type, and the possible value. In line 3, the dataset is created, i.e. the object in which the data mining process will be executed. If the actual runtime data or the data generated by the prototype is not being used, it is still possible to create the dataset via a simulation. In this case, the first step of the process is to generate the dataset (simdata) from a probability distribution of choice based on domain information[4]. For the BSN, we apply the Gaussian distribution to generate the data for most of our variables[5]. In line 4, we begin to visit each CGM node in the CGM tree (goal or task) and verify via data mining how the CGM nodes and its variables interact to successfully execute the parent tasks/goals, that is, which variable configurations of the lower tasks lead to system's goals fulfilment and which do not. We run through the CGM using a post-order depth-first search (DFS). In line 5, we apply Apriori [31] as an association rule method to detect correlated contextual variables, which are stored into *varlist*. The adoption of Apriori in our approach is paramount to optimize the search space of all possible variables that could be part of an arising world predicate (line 6). Apriori helps to isolate the only variables involved in a particular routine. Thus, to discover world predicates in this particular area, there is no need to carry unrelated variables to the next mining routine, bringing unnecessary complexity to the process. After defining the search space

[4] In case runtime data or data coming from similar systems is available, this step can be skipped.

[5] It should not be a threat to validity since there are plenty of well known probabilistic distributions available in the literature [36] suitable to create different arranges of datasets, each one semantically related to the respective domain.

through the previous method, we apply a ruler method to quantify those correlations and generate the operationalization list (*operlist*), i.e., combinations of contextual variables and their values that potentially generate new world predicates. We use a classification algorithm [32] for this purpose (line 7). JRip is particularly useful to analyse resources represented by numerical contextual variables, like sensor measurements. Additionally, the current node's id is also aggregated into *operlist*. Through the aforementioned process, one will assist the CGM's update by simplifying the identification of the nodes in which contextual constraints shall be inserted.

In line 9, we create a context variables list which corresponds to a contextual syntax table formed by a world predicate id associated with the corresponding rules in *operlist*. In line 11, we update the simulated dataset by replacing any occurrence of the rule by its corresponding predicate id present in *contexts*. This routine significantly aids visualizing the results of further steps.

In lines 13 and 14, the CGM starts to be traversed again, verifying if the visited node has any associated context variable. If that is the case, the CGM is updated in line 15 and the world predicates ids are inserted into their corresponding node. This part belongs to the semantic association step of the data mining process and relies on human interaction to link the discovered contexts variables with the corresponding goals through the CGM.

The next step is meant to discover how different world predicate combinations influence the success or failure of a CGM task or goal. This routine can be executed in parallel with the previous one (line 15), since the input artifact here is the dataset itself, that is modified only in line 11. In line 19, a classifier routine runs through either J48 [37] or JRip for such purpose. This step, combined with the one in line 20, provides a quantitative analysis of the contextual variables interactions into the goal achievability, given the contextual constraints that need to be respected. This knowledge is useful to develop better adaptation plans, focusing on the system's most common and problematic operations.

The decision tree obtained after the data mining process carries most of the knowledge on how the context specification can be enhanced. Basically, the categories of decisions supported by the algorithm are represented in two major classes: i) contextual variables dependencies, and ii) relevance of the contextual variables. Figure 4 is an illustrative decision tree that represents the verification and classification of a given system (e.g. goal fulfilment, dependability attribute satisfaction, etc.) in presence of contextual variability. The parent nodes carry the contextual variables, while the leaf nodes represent the classification class for each path. The edges' labels represent the values of the contextual variable observed in the respective parent node. Using these concepts as background, we will verify the implications of the decision tree as follows:

– **Contextual variables dependencies:** The dependencies between variables are observed in every branch that links the root node to a leaf node, namely paths. Each contextual variable value, represented by a letter from 'A' to 'J', is linked to an adjacent variable value, in the same path, by an "and" connector. For instance, in path 3, which leads to a "True" outcome, we can verify a correlation between the context variables 1, 2 and 3 in the form: *Contextual Variable 1 == 'D'* ***and*** *Contextual Variable 2 == 'E'* ***and*** *Contextual Variable 3 == 'F'* $\Rightarrow$ *Class = 'True'*. Such interaction between contextual variables supports the detection of emergent world

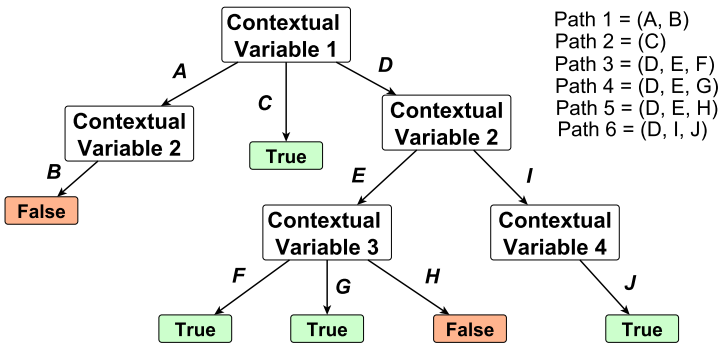

Fig. 4: Decision tree with contextual variables dependencies and their hierarchy

predicates that was unknown until the system's execution. The edges' labels can also display the world predicates or contexts that are related to the node's context variable, instead of showing the values of such variables.

– **Relevance of contextual variables:** Although our approach already filters the contextual variables that are, in fact, important to the fulfilment of a target quality constraint, there is still the possibility of ranking such variables, among the chosen ones, according to their relevance. According to the entropy and information gain theories [37], the higher the contextual variable is in the tree-like structure, the more relevant it is to the decision making process. Thus, it is possible to infer that the contextual variable 1 is the most important of all, followed by variable 2 in an intermediary layer and finally by variables 3 and 4, as less relevant. Although all these variables must be fulfiled, the relevance may be related to the criticality of eventual failures they are related to. If the analyst wants to quantify this relevance, it is sufficient to calculate the information gain of each variable. Such knowledge about the criticality of some variables can be useful to optimize the monitoring process by prioritizing the monitoring of some specific variables, as we may not need the others to fulfil every goal.

Regarding the dynamism of the models at runtime, different variables might be introduced during the system execution, however it will only be taken into consideration by the data mining process if it has a significant relationship with any world predicate or context. The CGM is supposed to describe the achievement of goals in face of contextual constraints. If it is noticed, after running our method, that an architectural change is required, it is clear that a remodelling process is also necessary. However, the mere introduction of variables does not imply changing the model. If the execution order of tasks and goals is essential, the model can be updated to a Runtime Contextual Goal Model, but for now it extrapolates the scope of the present work.

Figure 5 exemplifies the steps of our algorithm on the realization of task T1.3, further refined into tasks T1.31 and T1.32 in Figure 1, which is executed by the persistence module. Such a module is essential to the BSN goals, since it is responsible for storing the knowledge about the patient's health, that will be subject to the mining process in the future and will support the decision making in the adaptation process. We represent

the algorithm as five steps and the outcome of each step as alphabetic letters to represent the stages reached by the algorithm.

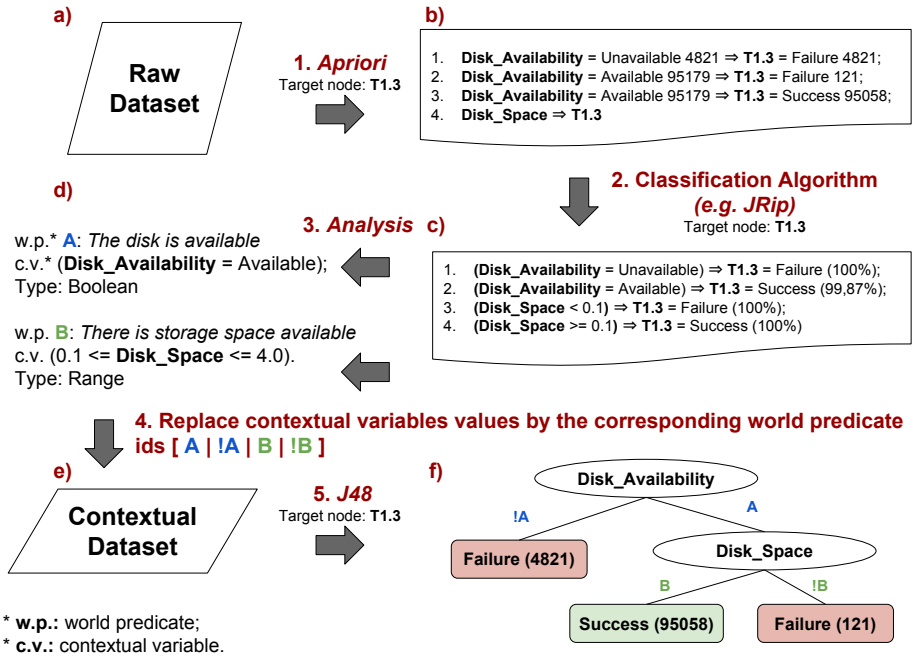

Fig. 5: Data mining process for the persistence module described in Figure 6

As a first step, represented by (1) in Figure 5, the Association Rule method is applied on the dataset generated in stage (a) of Figure 5 in order to discover the correlation between contextual variables. It would be possible to verify, as described in stage (b), the variables like *Disk_Availability*, related to the leaf task T1.31, and *Disk_Space*, related to the task T1.32, work together to successfully realize their parent task T1.3. The application of the Apriori method in step (1) enhances the efficiency of the JRip procedure in step (2), filtering only the attributes that are somehow related to the target node. Noticing the clear relation between the two variables and task T1.3, the three variables (T1.3, *Disk_Availability* and *Disk_Space*) are isolated and the ruler method JRip is applied over the specific attributes on step (2). The returned rules detail how these two variables relate with each other and how they cooperate to complete task T1.3. A possible rule generated by JRip could be translated to: “every time the storage device is unavailable, the disk space is unknown” or “every time the disk space is below 100MB, task T1.3 fails”. The rules and relations that imply tasks’ or goals’ failures are counted during the process, generating a probability report as presented in stage (c), pointing out to the likelihood of an event to occur. After the CGM traversal, the data analyst has enough material to scrutinize and come up with the most sensitive variables

and possible bottlenecks of the system. After the analysis in step (3), the contextual syntax table, stage (d), is produced to express the relation types of the newly selected contextual variables. To illustrate, a record of a syntax table could be expressed as: B (*world predicate id*); Boolean (*context type*); "There is storage space available" (*world predicate*); **If** Disk_Space $\geq$ 100MB **and** Disk_Space $\leq$ 4GB **then** B is true, **else** B is false (*contextual variables operationalization*). The world predicate ids are placed by the analyst in the corresponding refinements of the CGM. Meanwhile, in step (4) the rules that represent a given context in the dataset are also replaced by the respective world predicate id. For example, for every record of the simulated dataset where the disk has no storage available, i.e., the Disk_Space value is below the defined threshold, the corresponding value is replaced by the world predicate tag '!B'. The same happens to every other contextual variable present in the context variables list. At last, having a dataset only in terms of world predicate ids, stage (e), it is possible to advance to step (5) and apply J48 to create a decision tree. The step (5) of Figure 5 supports the analysis of the possible context variants behaviour in face of different aspects of the system. The decision tree in stage (f) indicates how the contextual variables Disk_Space and Disk_Availability relate to the world predicates 'A' and 'B' to fulfil the target node T1.3. By analysing the decision tree in stage (f), one can notice a red leaf box represented by *Failure (4821)*, which signalizes the fail rate of T1.3 due to unavailability of the storage device, including almost 4.82% of the total amount. The second red box, *Failure (121)*, indicates that the task T1.3 fails due to lack of storage space in approximately 0.12% of the executions. On the other hand, the green box labeled as *Success (95058)* shows the success rate of task T1.3 when both contexts are satisfied, representing approximately 95.06% of the records.

Throughout the example, it can be noticed that the data mining revealed that some variables that were ignored at design time are, in fact, (i) involved in the operationalization of a given context. Moreover, (ii) the range of values that each contextual variable may assume can also vary in time. To illustrate, consider the following scenario: an analyst defines, at design time, that the *storageProtocol* context (*sP*) is composed of the world predicate "the database is available". He also defines that the operationalization of such world predicate is based on the contextual variable "Disk_Availability" having a "TRUE" value. Although this specification is valid, it is incomplete, as the analyst observes via data mining that, (i) the contextual variable "Disk_Space" is also crucial to define whether the storage process will be successful. This fact was only revealed because such a variable was already defined at design time, making the rule "Disk_Availability = TRUE **and** Disk_Space $\leq$ 100MB" => "Fail" explicit in the analysis. Now, the variable "Disk_Space" becomes a contextual variable in the operationalization of a new world predicate: "the database has storage space available". Additionally, (ii) through the data mining analysis, the analyst notices that the values fixed for this context at design time are not valid for every case, as each data package might have a different size at each iteration. For instance, fixing the minimum free space of 100 MB before planning a contingency action might not be the best strategy for a dynamic communication environment. This example shows the effectiveness of our approach in identifying relevant changes in the variable itself, besides the variability in their values/combinations.

Figure 6, illustrates the refinement of task T1.3 (an excerpt of Figure 1), which is related to the storage process of the BSN sensors information either on a database or on a file system storage.

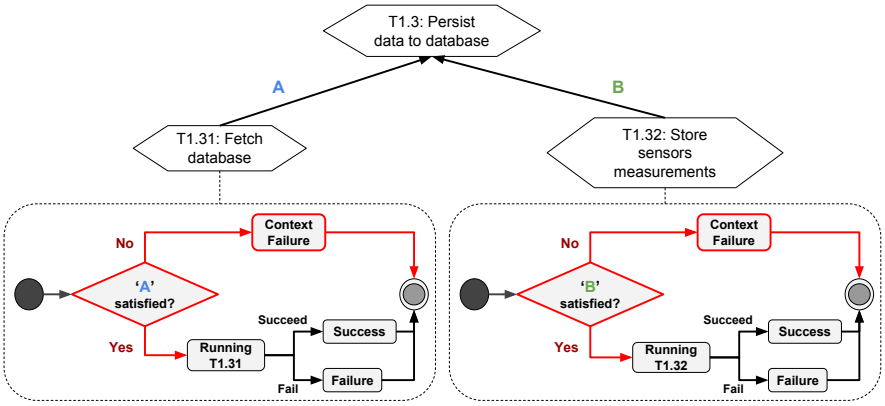

Fig. 6: BSN's CGM excerpt with associated world predicates (adapted from Figure 1)

4 Evaluation

4.1 Research Questions

In this section we evaluate our approach by means of the Goal-Question-Metric (GQM) methodology [38]. The questions that are relevant to evaluate our approach are divided into two major parts: one regarding the algorithms used in the data mining process itself and the other regarding the efficacy of the method as a whole. Following the research goal guidelines recommended by [38], we present the description of our evaluation goals in the GQM structure. As described in Table 4, the purpose of goal 1 is to analyse our data mining algorithms w.r.t to their reliability regarding the precision and recall rates. The purpose of goal 2 is to analyse how relevant and impactful the contexts identified by means of our approach are. The research questions are detailed in Table 5.

Goal 1	Goal 2
Analyse the data mining algorithms	**Analyse** our method
for the purpose of evaluating their reliability	**for the purpose of** identifying new contexts
with respect to precision and recall	**with respect to** their relevance and impact
from the viewpoint of a researcher	**from the viewpoint of** a researcher
in the context of of a BSN prototype.	**in the context of** of a BSN prototype.

Table 4: Definition of evaluation goal 1 and goal 2

Goal 1: Analyze the classification algorithms of our context mining process.	
Question	Metric
1.1 How reliable are the prediction models provided by the selected classification algorithms?	Precision and Recall Rates

Goal 2: Analyze how relevant and impactful the newly identified contexts are.	
Question	Metric
2.1 Can our approach identify new world predicates related to dependability for the BSN?	Amount of relevant world predicates
2.2 What is the impact of the newly identified contextual variables and relations on the BSN goals' satisfaction?	% reliability satisfaction for levels of patient health risks

Table 5: GQM plan

4.2 Experiment Planning

I - Variables: With respect to *Goal 1*, in which we analyse the quality of the prediction models generated by the classification algorithms, the dependent and independent variables can be seen from two points of view. In the first one, the independent variables are found in the data extracted from the executing system, i.e, they are domain related variables that represent the states of the BSN system. The dependent variables are the target classes of the classification. In our experiment, we have one target class, which stores the patients health risks as object of the prediction. In the second perspective, we can say that the correlation between the monitored variables and the target class is an independent variable as well, while the measured precision and recall are dependent variables. This is quite straightforward, since a fully randomized dataset leads to low levels of precision and recall, while a dataset with related variables tends to provide more insightful prediction models.

As for the *Goal 2*, although one of the objectives of our method is to reduce the human interaction in the context elicitation loop, the domain knowledge level of the stakeholder can still be considered an independent variable. This is often reflected in the number of context variables identified as relevant during the process. Accordingly, the identified context variables are taken as dependent variables. Additionally, the tree generated by our method is dependent of the quality constraint thresholds defined by the stakeholder. We dissect this idea in the next topic.

II - Hypotheses: We evaluate our approach on the BSN running example, where the *healthRisk* is taken as the initial contextual variable and acts as a trigger to activate the adaptation plans. Our approach intends to derive new contexts from the combinations of

the specific contextual variable and the system goals. The patient status can be defined by processing the data gathered from sensors in different configurations.

For the operationalization of the patient status, there are three possible risk states: *low*, *normal* or *high*. The objective of the BSN is to adjust the sampling rate (data gathering of patient's measurements) based on the patient's status and its required reliability, according to [21,19]. The initial adaptation rules are defined by the following hypotheses: if the patient presents a high risk status, his vital signs shall be checked every minute in order to ensure his safety. If the patient status is processed as normal risk, the sensor data collecting rate must be five minutes. At last, if the patient is in a low risk state, a verification of his vital signs every fifteen minutes is enough to guarantee his safety.

The operationalization of each patient's health risk state has one or a few possible resource configuration behind it. Following reported data[6], we set the minimum reliability level required for each patient state measurement as: 95% for low risk, 96% for normal risk and 97% for high risk. Considering the reliability reported data ranges from approximately 95% to 99%, our set of reliability ranges means that at most 5% probability of failure for the resources is acceptable for the low risk patient, at most 4% probability of failure is acceptable for the medium risk, and at most 3% probability of failure is acceptable for the high risk patient. We should note that our choice for such reliability ranges is based on the reported reliability ranges and should not represent a threat to validity since it could be adjusted to vary in accordance with the domain expert requirement. In the BSN running example, such resources are reported as features following a Software Product Line approach. For example, a configuration < !SP02, TEMP, ECG, ACC, Pos, !Oxy, Fall, PulseRate, Temperature >, that has 95.31% of reliability, means that all resources and variables but SPO2 and Oxygenation are present and its reliability level would suffice to satisfy the quality constraint of a patient in a low risk state. However, it wouldn't meet our threshold for the required reliability level of a normal or high risk state patient.

Summarizing, the hypotheses we want to test experimentally are: i) the supervised learning algorithms in our method provide reliable classifications, and ii) our method is able to identify impactful and relevant world predicates that support the development and evolution of a more reliable BSN.

III - Design: The first step in designing our experiments is setting the prototype resources and the operation logic, and executing it. The operation range of each resource is analysed and taken as parameter to support its simulation. Then, following the adequate distribution for each variable, the database that will be used on data analysis stage is created. For instance, one could use the Gaussian distribution to simulate the individual's heart beat data, mimicking a working pulse sensor over time.

Regarding the *Goal 1*, before starting the data mining procedure, we must firstly verify whether the techniques applied are adequate and the data preprocessing was done correctly or not. We focus on three pillars: (i) accuracy, that represents the prediction model efficiency in correlation to the outcome with the attributes in the provided data; (ii) reliability, that represents how well the prediction model will behave in face of

[6] https://code.google.com/p/spl-model-checking/

different datasets; and (iii) usefulness, that tells the applicability of a given method in extracting useful information from a dataset. Summarizing, the data mining stage begins with the validation of both, the generated data and the data mining techniques that we should apply. To measure the accuracy of the data mining process, we take into consideration metrics as precision and recall. As means to quantify the reliability of the answers provided we executed the prototype ten times, generating ten different datasets to be analysed. In order to measure usefulness, we applied five known classification methods (J48, JRip, One-R, Decision Table and Bayes Net) for each dataset.

In *Goal 2*, our method is applied aiming at validating our second hypothesis. The contextual variable value associated to each resource arises from a simulation of 100,000 records. A record is a row of the dataset that contains the resources values in a given moment, like a snapshot of the system status at the time. Such resource values comprise those elements enough to identify the patient's health status. In a nutshell, a record describes the state of the system and environment in a given moment. Each record is composed by a set of different vital signs measurements of a fictitious person based on a well known patient guide [39], following the work reported in [21] and valid sets of resources configurations available online[7]. The undesired behaviours of the simulated sensors were classified in two groups: unavailability (unexpected interruption of the component's operation) and outlier detection (monitored data out of expected range). This classification helps the mining process to identify how the contexts influence the occurrences of undesired behaviours.

Still related to *Goal 2*, we chose the output of a dependability analysis to collect the initial quality constraints as a preliminary construction of our experiment. Although the input data of our experiment regarding the reliability of each component had been gathered from a model checking process, and supported the definition of the acceptable ranges of operation for each patient health risk, this does not mean that it is the only way to define a quality constraint, neither the only attribute that could be analysed. On the BSN, the major quality constraint is reliability, where each set of resources configurations provides a specific reliability level.

4.3 Experiment Operation and Data Analysis

To demonstrate the applicability of our approach, we developed the BSN prototype, after building its CGM previously presented in Figure 1, having only the *healthRisk* as its initial elicited context, which is used to express the current patient status. The prototype was developed in Python and the experiments were processed in an Intel Core i7 5500U, 2.4GHz, 6GB DDR3. The prototype assisted in simulating the data sent by the sensing devices to the processing unit. The processing unit works as an autonomous controller, processing the simulated measurements defining a health status according to the policies. The adaptation plans adopted after the processing step are sent to the effectors, that manages the user's vital signs sampling rate according to his status. Weka [40] was the selected tool to assist the execution of our classification algorithms. For the purpose of replicating our evaluation, all the artifacts of this experiment are available at a GitHub repository[8].

[7] https://code.google.com/p/spl-model-checking/

[8] https://github.com/ArthurJRF/IST

Goal 1: Analyze the classification algorithms of our context mining process.

Question 1.1: How reliable are the prediction models provided by the selected classification algorithms? We chose the operationalization of the *healthRisk* context as the prediction object for the classifying process. Each dataset has one hundred thousand records. For this test, the chosen contextual variables adopted to represent the record was: <*Oxygenation*, *Pulse Rate*, *Temperature*, *Position*, *Fall*, *Patient State*>. The training and testing phases were based on a ten fold Cross-Validation, a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it. As we can see by the results demonstrated in Table 6, all tested methods performed well in terms of precision and recall. Although the adopted algorithms (J48 and JRip) performed very similarly in terms of precision and recall in comparison with the Decision Table and Bayes Net, the J48 and JRip provide us fundamental tools for our method, i.e. the decision tree and the operationalization rules. The false-positive(fp) and false-negative(fn) results are described in terms of precision ($tp/tp + fp$) and recall ($tp/tp + fn$).

		J48	**JRip**	One-R	Decision Table	Bayes Net
Precision	Mean	**0.969**	**0.969**	0.850	0.969	0.969
	StdDev	**5.6765E-04**	**5.1640E-04**	1.595E-03	5.6765E-04	5.1640E-04
Recall	Mean	**0.968**	**0.968**	0.828	0.968	0.968
	StdDev	**5.6765E-04**	**6.7495E-04**	1.912E-03	5.6765E-04	6.7495E-04

Table 6: False positive and false negative analysis of the generated prediction models

The minimum support and minimum confidence level defined for Apriori algorithm was 0.55 and 0.9 respectively. Picking the appropriate values for support and confidence is a tricky job, since they depend on the domain knowledge to be optimized. Since we are dealing with a medical application, we empirically established a balance of support and confidence value in order to get an adequate set of new rules. For instance, if the support and confidence values are too high, very few or no rules will be returned by the association rule method, and one might lose useful insights. On the other hand, if the values are low, the algorithm will probably return many rules, however it is likely that the majority of them will not aggregate any helpful information.

Goal 2: Analyze how relevant and impactful the newly identified contexts are.

Question 2.1: Can our approach identify new world predicates related to dependability for the BSN? Figure 7 presents the final CGM, i.e. the BSN with new context constraints discovered in the process. This last CGM is an improved version of the CGM presented in Figure 1, in terms of completeness and correctness. Apart from the initial context hR (*healthRisk*), previously defined in the earlier version of the CGM, the new elicited contexts are: dS (*dataSensing*), cM (*communicationMethod*), sP (*storageProtocol*), sR (*systemResource*), and oP (*operationalizationProcess*). As depicted in Figure 8,

the underlying world predicates are those from WP1 to WP19, and they were related in terms of logical connectives.

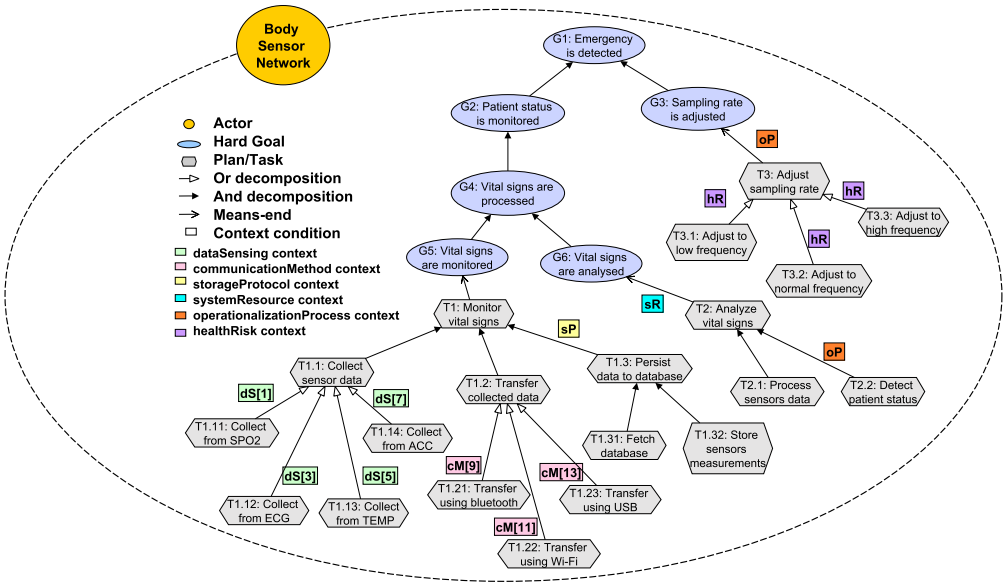

Fig. 7: CGM for the BSN system with detected contexts

Question 2.2: What is the impact of the newly identified contextual variables and relations on the BSN goals' satisfaction? In Figure 9 we report the results on the percentage of patients' health risk state that are either satisfied or not considering the elicited contexts for the BSN following our method for 100,000 patient records where 8.8% of the records fall into the high risk profile, 25.7% of normal risk and 62.8% of low risk. Moreover, 2.7% of the records identified critical operational failure in the system, which would render a crash of the BSN for all patients' health risk readings. Figure 9 also shows the non-identified state of some quality constraints, regarding its satisfaction or violation. The amount of non-identified states corresponds to the 4.62% of the total executions and it is related to the uncertainty of the prediction model, inherent to the data mining method.

Through the decision tree presented in Figure 10, as an outcome of the J48 algorithm, we are able to see the previous analysis from a different perspective, aggregating some knowledge that was not explicit before. Although our method already filters some variables that are involved in the decision making process and plot, in the decision tree, only the relevant ones, it is possible to go even further and create an importance rank among the plotted contextual variables. The most straightforward approach is to measure the information gain or gain ratio of each context variable and identify the variables that impact the most the classification of the records into the target class. By

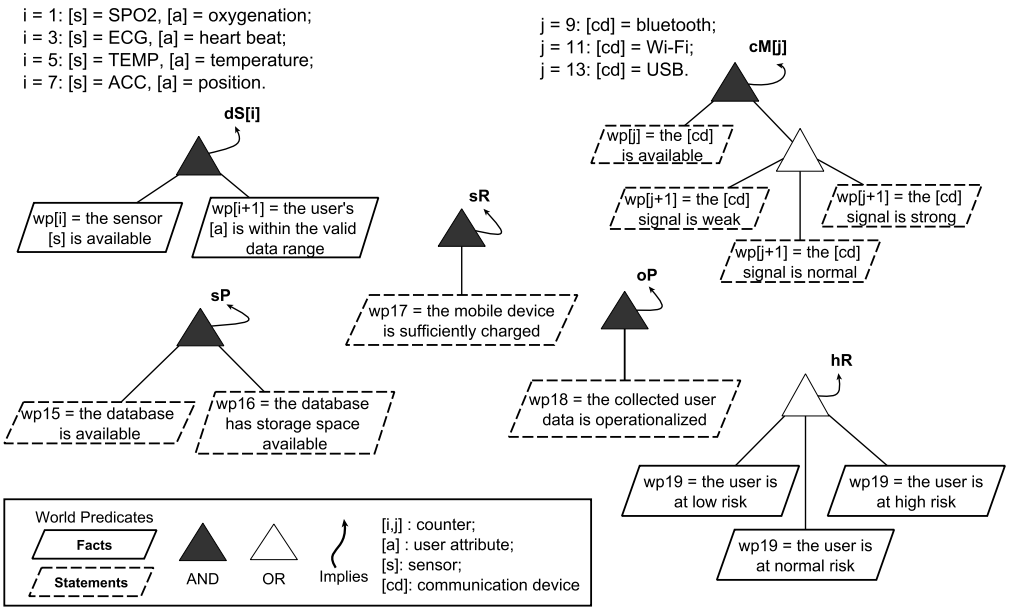

Fig. 8: Operationalization of each verified context

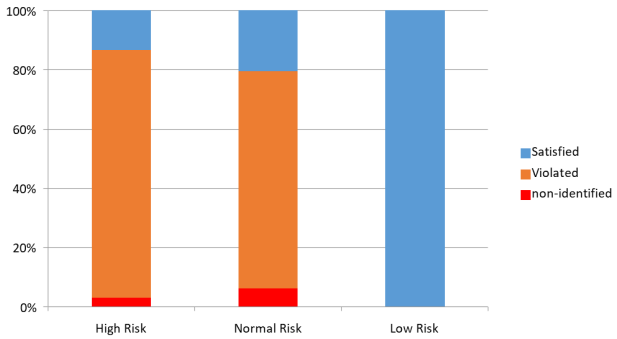

Fig. 9: Impact on the BSN goal satisfaction for patient's high, normal and low risk states under the contexts discovered through our data mining approach over 100,000 records.

the definition of the pruning process, the variables with higher score are closer to the root goal [37].

Figure 10, the orange box, labeled *Oper. Failure*, illustrates an unavailable context information, indicating that the system did not have enough information to characterize the contextual variables and assemble a world predicate as one of the three possible patient states defined for the *healthRisk* context. The leaf nodes were colored differently to distinguish the context configurations that, for a specific patient risk, lead to the ful-

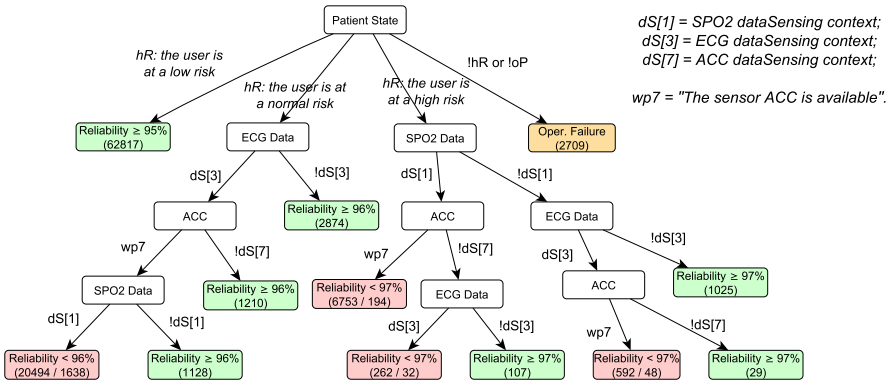

Fig. 10: Decision tree displaying the influence of some contextual variables in the achievement of quality constraints. Consider the following rules: a) dS[i]: "the sensor [i] is available" **and** "its data is within a valid range". b) !dS[i]: "the sensor [i] is unavailable" **or** "the data is out of a valid range". c) "the sensor is unavailable" $\Rightarrow$ "the data is out of a valid range". d) "the data is out of a valid range" $\not\Rightarrow$ "the sensor is unavailable". The structure of each context and associated world predicates is defined in Figure 8.

filment of the reliability threshold (green) from those that do not achieve the required level (red). The numbers in the leaf nodes refer to the amount of records found in the dataset in such context. The nodes with two numerical values, refer to the true positive amount (node's left-hand side) and false positive amount (node's right-hand side) generated by the prediction model. The false negative values are related to the amount of non-identified states mentioned before.

4.4 Interpretations of Results

Considering that we are dealing with a medical system, the low rates of false negative and false positive are crucial for a good classifier. With respect to the *Goal 1 - Question 1.1*, by running the classification routines as a preliminary validation, we verified a very low rate of false positives and negatives, proportionally speaking. We also can state that we have achieved satisfactory marks (the worst prediction model has precision and recall values above 96%), considering the non-deterministic property of some variables. Such marks indicate that the utilization of these data mining routines is significantly more valuable than a random guess or a naïve approach to extract information from the dataset. Therefore, our hypothesis that the selected supervised learning algorithms provide reliable classifications was validated in the current experimental conditions.

Analysing the results of the experiments related to *Goal 2 - Question 2.1*, our method supported the identification and operationalization of 6 contexts, composed of 19 different world predicates that were not present in previous published results of the BSN [19,21]. By detecting these new contexts and the corresponding world predicates, we hope to support the stakeholder, e.g. the system engineer, to anticipate the development of adaptation policies that cope with such contexts. This potentially avoids failures that come from context uncertainty, reducing maintenance and software evolution costs.

Finally, concerning the experiments of *Goal 2 - Queston 2.2*, results depicted in Figure 9 show the alarming fact that, without the awareness about the unveiled contexts raised by our method, only the low risk profiles would be completely satisfied, while almost 84% of high risk profiles and 73% of normal risk profiles would be violated (not attended). In other words, the high percentage of violations on both high and normal risk states allows us to infer that our elicited contexts do have a significant impact to identify critical health status, which was not previously taken into consideration on the BSN works. Moreover, adopting the theory around the pruning process which states that the variables with higher gain scores are closer to the root goal, it is possible to infer that following the *Patient State* context variable, the variables related to the information gathered by some sensors are the most relevant. More specifically, *ECG Data* and *SPO2 Data* are fundamental to define whether the reliability constraint of a given measurement will be respected, given the height in which they appear in the decision tree, as depicted in Figure 10.

4.5 Threats to Validity

Construct validity – In order to provide a reliable and sound data input for our evaluation, we relied on a reported sound exemplar (BSN) and its published available data. Despite all the care we took to avoid the generation of unrealistic data, we cannot guarantee that the generated data will perfectly represent a real case scenario. Further studies must be conducted to verify such representation.
Internal validity – In our evaluation based on the published information of the BSN, we were able to both, model the contextual goal model that reflects the BSN architecture goals, and vary the sensed data and perceive a significant impact on the outcomes. Therefore, the modelling structure showed itself efficient to adequately evaluate our approach and identify new relevant contextual variables for the BSN. However, unveiling all contextual variables involved in a system's operation is non-deterministic, which could represent a threat to the completeness of the elicited contexts and consequently to the overall dependability of the system.
External validity – Although our approach is not tailored to be domain specific, we do reckon the limitation of the evaluation since it was applied in the specific case of the BSN. Further evaluation of the approach must be performed to evaluate the actual applicability of our approach for generalization purposes.
Conclusion validity – All the input and output data have been reported and are available in the repository links provided in Section 4.3. In our experiments, well known statistical techniques were applied. The generated input information for the patients record to exercise in the BSN stem from real-life data of valid resources configuration set. Then, using well-established probability distributions, such resources configurations were simulated into 100,000 patient records. The reproducibility of our results are, therefore, constrained to the randomness of the data generated by the probability distributions adopted.

5 Related Work

As mentioned before, an early identification of possible context and the quantification of their impact on the system's behaviour is crucial to support the development of de-

pendable software. However, the lack of information at design time makes the identification of such contexts a great challenge for the software engineering of self-adaptive systems. In the scope of our work, such a challenge is tightly related to the concept of uncertainty, that begins at the elicitation stage and extends itself to runtime through the monitoring of the current context. Throughout this section, we enlist most work related to ours, regarding the challenges in elicitation of relevant context, dependability, specification, and adaptation of SAS to contextual changes, as well as uncertainty modelling and management in self-adaptive systems.

Context elicitation: With the creation of systems that operate in dynamic environments, the elicitation of contextual requirements has become a concern. A framework for requirements analysis that accounts for individual and personal goals and the effect of time and context on personal requirements was proposed by Sutcliffe *et al.* [13]. Observing the growing complexity of software systems, and the difficulty in eliciting requirements on the running environment, Keller [12] proposes an introductory overview on contextual requirements elicitation for software systems. A few years later, Sutcliffe and Sawyer [11] developed a road map for future requirements elicitation. Although they do not focus specifically on contextual requirements, they brought the notion of *unknowns unknowns* to the requirements elicitation process. Knauss *et al.* [2] contribute with a study on how to derive contexts from stakeholder needs. The authors explore the usefulness of several existing elicitation techniques for the identification of contextual requirements at design time. Hong *et al.* created a methodology for the elicitation of requirements in context-aware applications [3]. The proposed method links context-awareness features with the target context by capability matching. Gómez *et al.* [41] present APP STORE 2.0, an app store which applies data mining techniques to exploit crowdsourced information at runtime aiming at the elicitation of requirements, and improvement of the overall quality of the delivered service. Our proposal considers that the elicitation of new contextual variables and relations, at design time, can reveal possible uncertainties to the adaptive system, i.e., disallowed behaviours latent in the design that can jeopardize the requirements fulfilment at runtime.

Specification and adaptation to contextual changes: Ali *et al.* proposed a framework [4] that, starting from contextual requirements, involves reasoning and goal modelling to support the adaptation of a system to different contextual conditions. The choice and prioritization of the adequate goal model variant is a difficult task for complex systems, which motived us to use data mining in this sense. Villegas *et al.* present DYNAMICO [42], a reference model for governing control and context relevance in self-adaptive systems, in which they have defined, discussed, and implemented the preventive approach in the context of self-adaptation. We verify in our work that a similar analysis can be done, still at design time, to identify the resources and correlated contexts that, while available, guarantee the satisfaction of the probabilistic requirements.

Uncertainty definition for self-adaptive systems: Esfahani and Malek developed a study about uncertainty in self-adaptive systems field [8] claiming that, although uncertainty is sometimes taken as a second-order concept, it is not possible to remove uncertainty of a system by focusing exclusively on its normal behaviour. The work contributes with a classification and detailed description of several software aspects that can be considered as sources of uncertainty such as uncertainty in the objectives, uncertainty due to model drift, uncertainty in contexts, etc. Mahdavi-Hezavehi *et al.* [7]

propose a classification framework for architecture-based approaches tackling uncertainty in self-adaptive systems with multiple quality requirements. The work helps us to understand the current state of research regarding uncertainty. Another work that classifies and describes uncertainties in the context of adaptive systems was proposed by Ramirez *et al.* [6]. Whittle *et al.* claim that a more rigorous treatment of requirements explicitly relating to self-adaptivity is needed, more specifically the uncertainty-related aspects [43]. They present RELAX, a requirements language to explicit the presence of uncertainties in self-adaptive systems. Cheng *et al.* [44] take a step further and combine the RELAX specification with goal modelling to develop requirements of an adaptive system. Additionally, they use a threat modelling variation to explore environmental uncertainty factors. The aforementioned research give us some specific targets when the subject is uncertainty mitigation. Our work corroborates the fact that the use of data mining is a sound alternative to guide analysts to anticipate such uncertainties, avoiding possible time and space limitation, inherent of the combinatorial exploration in complex systems. Additionally, the initial domain knowledge that is used to specify the expected operational ranges for some contexts is often revealed as detached from the reality. The data mining process aligned with a continuous CGM update allows the prompt identification of contextual changes and, therefore, supports the adaptation of the system to environments that were initially unknown.

Dependability in context-based systems: From the dependability perspective, the closest in nature to our work combines contextual requirements and goal models: Mendonça *et al.* [16] propose a method to capture contextual failures and use that to enrich the representation of dependability requirements. The approach mitigates the assumption about the certainty of success of a task to reach its goal by adding contextual information about the quality of alternative tasks. However, due to the combinatorial if-then-else rules required from expert domain knowledge for fine grained individual analysis of contexts, the approach requires too much effort from the domain expert to guarantee an accurate context analysis. Differently from our work, the following research do not consider the context elicitation as a means to reach dependability, however they help us in delineating the scope of our work, considering the research gaps in dependability assessment for adaptive systems. Grassi *et.al* [45] present an approach that supports the assessment of performance and dependability attributes through a model transformation chain that maps a "design oriented" model to an "analysis oriented" model. Mahdavi-Hezavehi *et al.* [15] proposed a systematic literature review concerning to architecture-based methods for handling multiple quality attributes (QAs) in SAS. The authors say that performance and cost are the most frequently addressed set of QAs. Lemos *et al.* [46] survey on the challenges in the provision of assurances for SAS, the majority of the them are caused by the high degree of uncertainty introduced by runtime changes.

Tackling uncertainty at design time: A few works propose to deal with uncertainty still at design time. Among them, Horkoff *et al.* present an iterative methodology that guides the resolution of uncertainties necessary to achieve desired levels of goal satisfaction, assuming the model as the source of such uncertainty [47]. Hassan *et al.* created a method that is also worth mentioning [48]. Their method consists in allowing designers to make explicit links between the possible emergence of undesired surprises, risks and design trade-offs. 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. Our major contribution in comparison to other design time approaches is the use of data mining, still at design time, to analyse unexplored relations between resources that could be possible sources of new contexts, verifying the impact of such contexts in the satisfaction of functional or non-functional requirements with a view to dependability attributes.

Tackling uncertainty with AI methods: Many state-of-the-art contributions which involves artificial intelligence for dealing with uncertainties at runtime are present in the literature. To tackle this unpredictability at execution time, Knauss *et al.* proposed ACon [24]. The framework is based on machine learning and data mining techniques and it is meant to provide an adaptation of contextual requirements at runtime. The framework aids a self-adaptive system to adjust itself to overcome unexpected context variability and infrastructure failures through a feedback-loop. Esfahani *et al.* propose a similar approach [49], it uses machine learning to make a feature-oriented adaptation, focusing on features instead of contextual requirements. They deal with uncertainty through control and runtime adaptation. Welsh *et al.* [50] merge the uncertainty mapping with runtime resolution for the realization of requirements-aware systems through REAssuRE. They include claims to support the reasoning over uncertainty.

Work by:	**Appl. Stage**	**Goal-oriented**	**Dependability-oriented**	**Context elicitation**	**Techniques applied**
Hong *et al.*, 2005 [3]	Design	No	No	No	Reasoning
Cheng *et al.*, 2009 [44]	Design	Yes	No	No	Goal modelling, RELAX, threat modelling variation
Grassi *et.al*, 2009 [45]	Design	No	Yes	No	Model transformation, reasoning
Villegas *et al.*, 2010 [42]	Design	No	No	No	Reasoning, feedback loops
Welsh *et al.*, 2011 [50]	Runtime	Yes	No	No	Claims
Esfahani *et al.*, 2013 [49]	Runtime	Yes	No	No	Machine learning
Horkoff et al., 2014 [47]	Design	Yes	No	No	Reasoning
Knauss *et al.*, 2014 [2]	Design	Yes	No	Yes	Combined requirements elicitation techniques
Mendonça *et al.*, 2014 [16]	Design	Yes	Yes	Yes	Goal modelling, reasoning
Hassan *et al.*, 2015 [48]	Design	No	No	No	Multi-dimensional what-if analysis
Knauss *et al.*, 2016 [24]	Runtime	No	No	Yes	Machine learning, data mining
Gómez *et al.*, 2017 [41]	Runtime	No	No	Yes	Data mining, path analysis, topic modelling, feedback
Present work	**Design**	Yes	**Yes**	**Yes**	**Data mining, goal modelling**

Table 7: Comparative table of the related work and their properties

Our method leverages the use of data mining, prototyping, and contextual goal modelling to assist the elicitation and correctness of relevant contexts related to dependability attributes, reducing uncertainty already at design time, adopting a preventive-like behaviour. Table 7 summarizes and compares the most important properties and characteristics of each related work. The *application stage* column depicts whether the application of the work is meant to be during design or runtime. This is useful, since we claim that the dependability assurance process should be built over a combination of design and runtime techniques. The *goal-oriented* column reflects on research that adopts a goal-oriented perspective. Adaptive systems need to use some type of modelling. Hence, this column gives an overview whether the work supports a goal oriented modelling, as we do in our work. A goal model solution is an alternative to tackle this requirement as long as it provides first class support for change. Our goal is to improve context elicitation. Although the machine learning technique might benefit other aspects, such as performance, they are out of scope of this work. The role of the *dependability-oriented* column is to compare the focus of the related work that encompass concepts of dependability. The last dimension, *context elicitation* is meant to compare our work to others that use traditional requirements elicitation techniques to elicit relevant contextual requirements.

6 Conclusion and Future Work

In this paper we have illustrated the applicability of a preventive approach that assists the unveiling and refinement of relevant contexts at early stage of software development, i.e., before the implementation and deployment of the final product. Avoiding a combinatorial exploration, the use of data mining methods over the system's prototype data was quite useful to reduce the search space formed by the countless combinations of environmental and computational resources. We evaluated the proposal on the previously published BSN study and, in the present work, we were able to verify a significant improvement on the completeness and correctness of the elicited contexts before the implementation stage. As expected, we noticed that the unveiled contextual variables were influential in the fulfilment of relevant system goals.

Our method exploits artificial intelligence to deal with unanticipated changes by detecting unforeseen contexts and context variability. For future work we intend to extend this work to include self-adaptation and software evolution [10]. We plan to quantitatively analyse, from a dependability and user satisfaction perspective, the impact and criticality of new world predicates and contexts. From an evolution perspective, we need to explore the potential of our method to support the architectural and operational improvement in order to meet the new contextual demands. Further research is also necessary to generalize the applicability of our approach, expanding it to other domains and preparing the requirement for *unknown unknowns*. Self-adaptive systems can highly benefit from this kind of analysis, having the adaptation guided by the reports generated by our method.

Acknowledgements

Arthur would like to thank CAPES-Brasil for the financial support– finance code 001. Genaína thanks CNPq for the partial support under grant number 306017/2018-0. Furthermore, a great appreciation to the Dependability Group from UnB's Software Engineering Lab (LADECIC) as well as to the anonymous reviewers for their invaluable feedback.

References

1. Muñoz-Fernández, J.C., Knauss, A., Castañeda, L., Derakhshanmanesh, M., Heinrich, R., Becker, M., Taherimakhsousi, N.: Capturing ambiguity in artifacts to support requirements engineering for self-adaptive systems. In: Joint Proceedings of REFSQ-2017 Workshops, Doctoral Symposium, Research Method Track, and Poster Track co-located with the 22nd International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2017), Essen, Germany, February 27, 2017. (2017)
2. Knauss, A., Damian, D., Schneider, K.: Eliciting contextual requirements at design time: A case study. In: 4th IEEE International Workshop on Empirical Requirements Engineering, EmpiRE 2014, Karlskrona, Sweden, August 25, 2014. pp. 56–63 (2014)
3. Hong, D., Chiu, D.K.W., Shen, V.Y.: Requirements elicitation for the design of context-aware applications in a ubiquitous environment. In: Proceedings of the 7th International Conference on Electronic Commerce, ICEC 2005, Xi'an, China, August 15-17, 2005. pp. 590–596 (2005)
4. Ali, R., Dalpiaz, F., Giorgini, P.: A goal-based framework for contextual requirements modeling and analysis. Requir. Eng. 15(4), 439–458 (2010)
5. Avizienis, A., Laprie, J., Randell, B., Landwehr, C.E.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Sec. Comput. 1(1), 11–33 (2004)
6. Ramirez, A.J., Jensen, A.C., Cheng, B.H.C.: A taxonomy of uncertainty for dynamically adaptive systems. In: 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2012, Zurich, Switzerland, June 4-5, 2012. pp. 99–108 (2012)
7. Mahdavi Hezavehi, S., Avgeriou, P., Weyns, D.: Chapter 3 - a classification framework of uncertainty in architecture-based self-adaptive systems with multiple quality requirements. In: Managing Trade-Offs in Adaptable Software Architectures, chap. 3Managing Trade-Offs in Adaptable Software Architectures, pp. 45–77. Morgan Kaufmann (01 2017)
8. Esfahani, N., Malek, S.: Uncertainty in self-adaptive software systems. In: Software Engineering for Self-Adaptive Systems II - International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers. pp. 214–238 (2010)
9. Gervasi, V., Gacitua, R., Rouncefield, M., Sawyer, P., Kof, L., Ma, L., Piwek, P., De Roeck, A., Willis, A., Yang, H., et al.: Unpacking tacit knowledge for requirements engineering. In: Managing requirements knowledge, pp. 23–47. Springer (2013)
10. Weyns, D.: Software engineering of self-adaptive systems: an organised tour and future challenges. Chapter in Handbook of Software Engineering (2017)
11. Sutcliffe, A., Sawyer, P.: Requirements elicitation: Towards the unknown unknowns. In: Requirements Engineering Conference (RE), 2013 21st IEEE International. pp. 92–104. IEEE (2013)
12. Keller, T.: Contextual requirements elicitation: An overview. In: Seminar in Requirements Engineering, Department of Informatics, University of Zurich (2011)

13. Sutcliffe, A., Fickas, S., Sohlberg, M.M.: Personal and contextual requirements engineering. In: Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on. pp. 19–28. IEEE (2005)
14. Yang, Z., Li, Z., Jin, Z., Chen, Y.: A systematic literature review of requirements modeling and analysis for self-adaptive systems. In: International Working Conference on Requirements Engineering: Foundation for Software Quality. pp. 55–71. Springer (2014)
15. Mahdavi-Hezavehi, S., Durelli, V.H.S., Weyns, D., Avgeriou, P.: A systematic literature review on methods that handle multiple quality attributes in architecture-based self-adaptive systems. Information & Software Technology 90, 1–26 (2017)
16. Mendonça, D.F., Ali, R., Rodrigues, G.N.: Modelling and analysing contextual failures for dependability requirements. In: 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2014, Proceedings, Hyderabad, India, June 2-3, 2014. pp. 55–64 (2014)
17. Cohene, T., Easterbrook, S.: Contextual risk analysis for interview design. In: Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on. pp. 95–104. IEEE (2005)
18. Filieri, A., Maggio, M., Angelopoulos, K., D'Ippolito, N., Gerostathopoulos, I., Hempel, A.B., Hoffmann, H., Jamshidi, P., Kalyvianaki, E., Klein, C., Krikava, F., Misailovic, S., Papadopoulos, A.V., Ray, S., Sharifloo, A.M., Shevtsov, S., Ujma, M., Vogel, T.: Software engineering meets control theory. In: Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. pp. 71–82. SEAMS '15, IEEE Press, Piscataway, NJ, USA (2015)
19. Pessoa, L., Fernandes, P., Castro, T., Alves, V., Rodrigues, G.N., Carvalho, H.: Building reliable and maintainable dynamic software product lines: An investigation in the body sensor network domain. Information & Software Technology 86, 54–70 (2017)
20. Yang, G.Z., Aziz, O., Kwasnicki, R., Merrifield, R., Darzi, A., Lo, B.: Introduction, pp. 1–53. Springer London, London (2014)
21. Rodrigues, G.N., Alves, V., Nunes, V., Lanna, A., Cordy, M., Schobbens, P., Sharifloo, A.M., Legay, A.: Modeling and verification for probabilistic properties in software product lines. In: 16th IEEE International Symposium on High Assurance Systems Engineering, HASE 2015, Daytona Beach, FL, USA, January 8-10, 2015. pp. 173–180 (2015)
22. Dey, A.K.: Understanding and using context. Personal and Ubiquitous Computing 5(1), 4–7 (2001)
23. Finkelstein, A., Savigni, A.: A framework for requirements engineering for context-aware services. In: In Proc. of 1 st International Workshop From Software Requirements to Architectures (STRAW 01). pp. 200–1 (2001)
24. Knauss, A., Damian, D., Franch, X., Rook, A., Müller, H.A., Thomo, A.: Acon: A learning-based approach to deal with uncertainty in contextual requirements at runtime. Information & Software Technology 70, 85–99 (2016)
25. Nunes, V., Mendonça, D., Rodrigues, G., Alves, V.: Towards compositional approach for parametric model checking in software product lines. In: International Workshop on Architecting Dependable Systems (WDAS'13). SBC (2013)
26. Knauss, A.: On the usage of context for requirements elicitation: End-user involvement in it ecosystems. In: 2012 20th IEEE International Requirements Engineering Conference (RE). pp. 345–348 (Sep 2012)
27. Hastie, T., Tibshirani, R., Friedman, J.H.: The elements of statistical learning: data mining, inference, and prediction, 2nd Edition. Springer series in statistics, Springer (2009)
28. Kotsiantis, S.B., Zaharakis, I., Pintelas, P.: Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering 160, 3–24 (2007)

29. Harman, M.: The role of artificial intelligence in software engineering. In: Proceedings of the First International Workshop on Realizing AI Synergies in Software Engineering. pp. 1–6. RAISE '12, IEEE Press, Piscataway, NJ, USA (2012)
30. Hussain, S.: Survey on current trends and techniques of data mining research. London Journal of Research in Computer Science and Technology (03 2017)
31. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile. pp. 487–499 (1994)
32. Cohen, W.W.: Fast effective rule induction. In: Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995. pp. 115–123 (1995)
33. Mendonça, D.F., Rodrigues, G.N., Ali, R., Alves, V., Baresi, L.: GODA: A goal-oriented requirements engineering framework for runtime dependability analysis. Information & Software Technology 80, 245–264 (2016)
34. Guimarães, F.P., Rodrigues, G.N., Ali, R., Batista, D.M.: Planning runtime software adaptation through pragmatic goal model. Data & Knowledge Engineering pp. – (2017)
35. Jureta, I., Mylopoulos, J., Faulkner, S.: Revisiting the core ontology and problem in requirements engineering. In: 2008 16th IEEE International Requirements Engineering Conference. pp. 71–80 (Sept 2008)
36. Casella, G., Berger, R.L.: Statistical inference. Duxbury/Thomson Learning, 2nd ed edn. (2002)
37. Quinlan, J.R.: Simplifying decision trees. Int. J. Hum.-Comput. Stud. 51(2), 497–510 (1999)
38. Basili, V.R., Shull, F., Lanubile, F.: Building knowledge through families of experiments. IEEE Transactions on Software Engineering 25(4), 456–473 (1999)
39. Beth israel hospital - mit database patient guide
40. Frank, E., Hall, M.A., Holmes, G., Kirkby, R., Pfahringer, B.: WEKA - A machine learning workbench for data mining. In: The Data Mining and Knowledge Discovery Handbook., pp. 1305–1314 (2005)
41. Gómez, M., Adams, B., Maalej, W., Monperrus, M., Rouvoy, R.: App store 2.0: From crowdsourced information to actionable feedback in mobile ecosystems. IEEE Software 34(2), 81–89 (2017)
42. Villegas, N.M., Tamura, G., Müller, H.A., Duchien, L., Casallas, R.: DYNAMICO: A reference model for governing control objectives and context relevance in self-adaptive software systems. In: Software Engineering for Self-Adaptive Systems II - International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers. pp. 265–293 (2010)
43. Whittle, J., Sawyer, P., Bencomo, N., Cheng, B.H.C., Bruel, J.: RELAX: incorporating uncertainty into the specification of self-adaptive systems. In: RE 2009, 17th IEEE International Requirements Engineering Conference, Atlanta, Georgia, USA, August 31 - September 4, 2009. pp. 79–88 (2009)
44. Cheng, B.H.C., Sawyer, P., Bencomo, N., Whittle, J.: A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. In: Model Driven Engineering Languages and Systems, 12th International Conference, MODELS 2009, Denver, CO, USA, October 4-9, 2009. Proceedings. pp. 468–483 (2009)
45. Grassi, V., Mirandola, R., Randazzo, E.: Software engineering for self-adaptive systems. chap. Model-Driven Assessment of QoS-Aware Self-Adaptation, pp. 201–222. Springer-Verlag, Berlin, Heidelberg (2009)
46. de Lemos, R., Garlan, D., Ghezzi, C., Giese, H., Andersson, J., Litoiu, M., Schmerl, B., Weyns, D., Baresi, L., Bencomo, N., Brun, Y., Camara, J., Calinescu, R., Cohen, M.B., Gorla, A., Grassi, V., Grunske, L., Inverardi, P., Jezequel, J.M., Malek, S., Mirandola, R., Mori, M., Müller, H.A., Rouvoy, R., Rubira, C.M.F., Rutten, E., Shaw, M., Tamburrelli, G., Tamura, G.,

Villegas, N.M., Vogel, T., Zambonelli, F.: Software engineering for self-adaptive systems: Research challenges in the provision of assurances. In: de Lemos, R., Garlan, D., Ghezzi, C., Giese, H. (eds.) Software Engineering for Self-Adaptive Systems III, vol. 9640. Springer (2017)
47. Horkoff, J., Salay, R., Chechik, M., Sandro, A.D.: Supporting early decision-making in the presence of uncertainty. In: IEEE 22nd International Requirements Engineering Conference, RE 2014, Karlskrona, Sweden, August 25-29, 2014. pp. 33–42 (2014)
48. Hassan, S., Bencomo, N., Bahsoon, R.: Minimizing nasty surprises with better informed decision-making in self-adaptive systems. In: 10th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2015, Florence, Italy, May 18-19, 2015. pp. 134–145 (2015)
49. Esfahani, N., Elkhodary, A.M., Malek, S.: A learning-based framework for engineering feature-oriented self-adaptive software systems. IEEE Trans. Software Eng. 39(11), 1467–1493 (2013)
50. Welsh, K., Sawyer, P., Bencomo, N.: Towards requirements aware systems: Run-time resolution of design-time assumptions. In: 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, KS, USA, November 6-10, 2011. pp. 560–563 (2011)