Modelling and Analysing Contextual Failures for Dependability Requirements

Danilo F. Mendonça
Universidade de Brasilia
Brasilia, Brazil
dfmendonca@gmail.com

Raian Ali
Bournemouth University
Bournemouth, UK
rali@bournemouth.ac.uk

Genaína N. Rodrigues
Universidade de Brasilia
Brasilia, Brazil
genaina@cic.unb.br

ABSTRACT

The notion of Contextual Requirements refers to the interrelation between the requirements of a system, both functional and non-functional (NFRs), and the dynamic environment in which the system operates. Dependability requirements are NFRs which could also be context-dependent. The meaning and the consequence of faults affecting dependability vary in relation to the context in which a fault occurs. In this paper, we elaborate on the need to consider the contextual nature of failures and dependability. Then, we extend a contextual requirements model, the contextual goal model, to capture contextual failures and utilize that to enrich the semantic of dependability requirements. We provide techniques to analyse and reason about the effects of contexts on failures and their consequences. This analysis helps evaluate the possible alternative configurations to reach goals from dependability perspective and, hence, take adaptation decisions. Finally, we demonstrate the feasibility and applicability of our approach on a Mobile Personal Emergency Response system.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specification

General Terms

Design, Reliability

Keywords

Requirements engineering, Dependability, Context

1. INTRODUCTION

Socio-technical systems provide and control a wide range of daily used services. Often, these systems are responsible for important and even critical requirements whose failures would cause undesirable or intolerable consequences. This requires developers to take dependability into consideration as a first class requirement which has social and technical elements and encompasses several attributes such as reliability, availability, safety, security and maintainability [3].

Stakeholders' requirements have essential role in adaptation. The ultimate goal of a system and its adaptation is to enhance the fulfilment of requirements. Non-functional requirements (NFRs) could act as quality measures for the different software configurations and alternative solutions to reach functional requirements. Dependability requirements is a specific kind of NFRs. A system can adapt in order to choose a configuration which is unlikely to lead to failure in meeting the requirements. Both kinds of requirements could be contextual. Requirements engineering has recently considered the effect of dynamic context on requirements, e.g., [1]. These approaches are not tailored to the peculiarities of faults and dependability.

Adaptation to maximize dependability traditionally relied on runtime data to perform adaptation actions capable of restoring the proper system behaviour when a failure occurs. An example of that is the notion of Live Goals meant as a driver for adaptation at runtime. Not meeting these goals triggers the need for adapting the system. Such failure could happen due to dynamic QoS of the composing parts of the system or due to new business needs [4, 6, 5].

Unfortunately, some systems cannot afford to wait for a failure to occur to perform adaptation as the consequences could be too severe or catastrophic in certain contexts. Also, historical data of past operations may not be available for the system and analysts to learn from. Hence, preventive and not reactive approaches must take place in order to decrease chances of failure to occur. Moreover, a proper balance between a dependability requirement analysis, which is based on domain experts, and reactive adaptation mechanisms, which are based on information monitored at runtime, would be needed to improve dependability.

In this paper, we tackle the context effects on systems dependability within two perspectives. Firstly, we consider the impact of context on the likelihood of a failure to occur, as some failures may be more likely to occur in certain contexts. These failures are expressed in terms of dependability attributes according to a failure classification scheme. We call this cause-effect relation as Contextual Failure Implications (CFI). Secondly, the consequence of failures may also be affected by the context. A failure, which is typically minor, may become a catastrophic failure in a certain context condition. We refer to this as Contextual Dependability Requirements (CDR).

As a technical contribution, we propose a modelling and analysis framework for contextual failures and dependable

SEAMS '14, June 2–3, 2014, Hyderabad, India

system requirements. The models offered by Goal-oriented Requirements Engineering (GORE), e.g., i* [18], Tropos [7] and KAOS [9], capture the socio-technical structure of a system and the a space of alternative solutions to reach stakeholders' requirements (goals) and their quality (softgoals). This makes GORE an ideal family of models to study dependability which encloses social and technical elements, and adaptation which requires variability. GORE has been already used to capture contextual requirements, e.g., the Contextual Goal Model (CGM) [1]. We extend the CGM by defining the causal effects among contexts and failures. We also propose automated analyses which utilize the model to assess dependability. We demonstrate the feasibility of our proposal with a case study for a Mobile Personal Emergency Response system (MPERS).

The paper is structured as follows: Section 2 presents the baseline and related work; Section 3 describes the proposal model, its formalization and the applicability; Section 4 presents an algorithm to validate the contextual dependability requirements against defined values of dependability attributes for system goals with underlying tasks in multiple contexts of operation; Finally, Section 6 concludes the paper and outlines our future work.

2. BASELINE AND RELATED WORK

2.1 Goal Modelling

Goal modelling is suitable for representing stakeholders intentions, both strategic and tactical. It allows the analysis of goals to finally achieve alternative sets of executable processes (tasks or operations) [18, 7, 9]. Some common concepts of a goal model are actors, goals, softgoals, decompositions, tasks and contribution links. Each system actor has its own interests defined as *goals*. A goal represents a partial state of the world an actor wants to reach. *Tasks* are the *operationalization* of actor's goals and they could be mapped to the execution of some operation by the corresponding system actors. Goals are crispy and they can be either satisfied or not. *Softgoals* are qualitative objectives for whose satisfaction there is not clear cut criteria. *Contribution links* identify the evaluation of goals and tasks against softgoals.

Goal models are commonly used for modelling and guiding systems adaptation. The reason for their popularity in this domain is mainly due to their ability to capture the different ways to reach the system goals and their ability to link between the software world (the tasks) and the stakeholders' world (their goals and quality requirements). Thus, they provide a platform to capture the rationale of adaptation by switching between its different tasks configurations to reach goals. In other words, they provide meaning to software adaptation from the business perspective.

2.2 Context and Goals

The environment in which a system operates could be dynamic. This is particularly true for new computing paradigms such as pervasive, ubiquitous and mobile computing. Context changes could affect the requirements of a system. Certain contexts could trigger a requirement and affect the applicability and the quality of a certain alternative or software variant designed to reach a requirement. From a GORE perspective, contexts can be defined as the partial state of the world that is relevant to an actor's goals [1].

The context analysis proposed in [1] provides constructs to analyse a context described at a high level of abstraction, e.g., "patient is not feeling well", to a formula of observable facts, e.g., 'temperature is high' and 'sweating'. Obviously, a fully-automated monitor will be able to only consider contexts which could be refined to a formula of facts and these facts should be verifiable based on data monitorable by some automated means like sensors, databases, and video or audio stream analysis. Other contexts could require a human perception, e.g., a caregiver can report the current status of a patient if the smart home or the ambulance do not have the required technology. The decision on the relevant contexts in a goal model is part of the requirements analysis activity. This is carried out in parallel with goal analysis by asking questions such as 'when do we need to activate a goal?' and 'when can we adopt a certain alternative to reach a goal?'. For contextual dependability, we could similarly identify relevant contexts by answering questions of the types 'when is this fault considered severe or could significantly impact system behaviour?'. The rationale here is similar to the way we analyse goals and decide the space of alternatives to reach a goal and the contribution to a softgoal. That is, it requires both expertise of requirements analysts and stakeholders who know the peculiarities of the domain.

As an extension to context analysis which considers only Boolean facts, we also consider fuzzy facts. The later means that their values vary in different degrees of membership for their intensity and quantity. Our proposed modelling of Contextual Dependability Requirements and Contextual Failure Implications proposed in Section 3.1 is based on contextual conditions that are formed by combination of facts formalized as predicates such as the battery level, the availability of a power supply and the number of sensors in use followed by their quantifier level and joined by AND/OR logical operators.

2.3 Dependability Analysis

The concept of dependability is related to dependence and trust as well as the ability of a system to avoid failures that are more frequent and more severe than certain threshold. Dependability encompasses the following quality attributes [3]: availability, reliability, integrity, safety and maintainability.

A holistic dependability specification has to include not only the software operation, but also the requirements for which that operation is meant. Requirements are an important factor to decide the acceptable frequency and severity of a software failure. Similarly, context is another factor in that decision. The frequency and the likelihood of failures are related to the dependability attributes of reliability, availability and integrity. Both likelihood and severity of failures are related to safety, as it defines the absence of catastrophic consequences on the users and the environment. In this stage of our work, maintainability is the only dependability property we have not considered.

At early project phases, hazard resolution may involve simply getting more information about hazards or generating alternative design solutions [11]. In our work, we have used a qualitative means to analyse the dependability to be delivered by goals of a certain system taking into account contextual effects. Our approach improves the understanding of systems fault-causality effect and the identification of

best approaches to reduce risk or even determine rates for safety or system level functional failure. Moreover, dependability requirement analysis cannot be accurately fulfilled without taking into account the context under which the system will operate.

Avizienis et al [3] proposed a failure classification taxonomy with four viewpoints characterizing failures. In our approach, we use two categories: domain and consequence. We use the domain category to distinguish *content* failures from *timing* failures:

- **Content failure**: When the content of the information delivered by a system task deviates from its specification
- **Timing failure**: When the time of arrival or the duration of the information delivered by some system task deviates from its specification

The consequence of failures enables the definition of failures' severity. Two limiting levels are predefined and other intermediary levels could be defined for each case:

- **Minor failure**: The harmful consequences of failures are limited or at most similar to the benefits provided by the correct operation of the system
- **Catastrophic failure**: The harmful consequences of failures are incommensurably higher than the benefits provided by correct operation of the system

More details on the complete failure classification can be found in [3]. In our approach, a part of this taxonomy is used to guide the definition of the classes of failures severities and therefore the required level of dependability for different system goals. It also takes part in the identification of which dependability attribute is related to each contextual failure occurrence.

2.4 Fuzzy Logic

The fuzziness of requirements has been discussed in the literature of requirements engineering for adaptive systems [12, 4, 6, 11]. Fuzzy words are considered to be closer to the natural language and mind-set of the stakeholders, including domain experts and analysts, involved at early stage of requirements elicitation and analysis [11]. This also applies to how we define contexts, failures and dependability.

The fuzziness consists of a range of values to be considered as 'around' a desired value. In other words, it allows a fuzzy description of what should be considered as a low battery, a high temperature, a strong signal, a reliable solution, etc. Also, as some context facts like the battery level, the temperature and the signal level of a cellphone are usually available as numeric values, membership functions are used to map these quantitative levels to the corresponding qualitative words, as presented in Figure 1. This step is part of fuzzy logic and is called *fuzzification*.

Fuzzy logic is suitable to express the causal relation of contexts and dependability attributes defined by fuzzy words, as it has a well established formalism and structure that includes the *fuzzification* and *defuzzification* through membership functions, logical operations able to combine inputs and rules used to define a crispy output given some fuzzy inputs [14]. When more than one rule is activated at the same time, the aggregation step will produce an output weighted

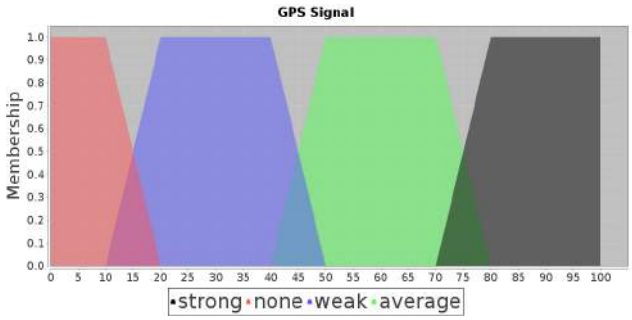

Figure 1: Fuzzification of the GPS signal level

by the resulting membership degree of each used input combination.

In our approach, fuzzy logic is used to tackle the fuzziness of the context facts and their effect on both dependability requirements and failures. Membership functions must be defined according to stakeholders preferences and the knowledge of domain experts and reliability engineers. Their definition will impact on the activation of rules used to evaluate a crispy output in terms of dependability that could finally be used as input to be validated against CDRs. Thus, appropriate effort should take place at this stage in order to improve the quality of the CFIs.

Other reasoning mechanisms rather than fuzzy logic could be applied to our framework. Nonetheless, we consider it suitable for representing the fuzziness of context facts and their relation to dependability attributes though its mechanism of rules activation and aggregation. Imprecisions in the definition of membership functions and rules could also exist on different approaches to correlate contexts conditions to a system failure. Therefore, fuzzy logic is a good option to utilize in the context of this work.

2.5 Mobile Personal Emergency Response

To demonstrate our approach, we take a Personal Emergency Response System (MPERS) scenario which is used via mobile devices (smartphones) and includes a network of sensors communicating through wireless technology [13]. Battery is a major restriction for this architecture, as patients may need the service anywhere, including areas without a power supply. Depending on the health condition of the patient, it could be essential to keep the system running without interruptions, i.e., the system availability must be high. Considering the existence of different alternatives for goals of the system, like the goal of identifying patients' location, the proper choice of which alternative to use must respect the balance among the different dependability attributes.

Figure 2 presents a full Contextual Goal Model (CGM) for the above scenario. The quality of several tasks in this system varies according to the context condition. Let us take as example the different ways of identifying patients' location; through cellphone towers triangulation, GPS or by direct voice call to the patient. According to the context, triangulation may be unreliable due to imprecision, a GPS could become unavailable inside uncovered areas and the voice call may not work if patient is in the middle of a heart attack. Ignoring the context effect on the dependability of these alternative solutions can result in an unsafe system,

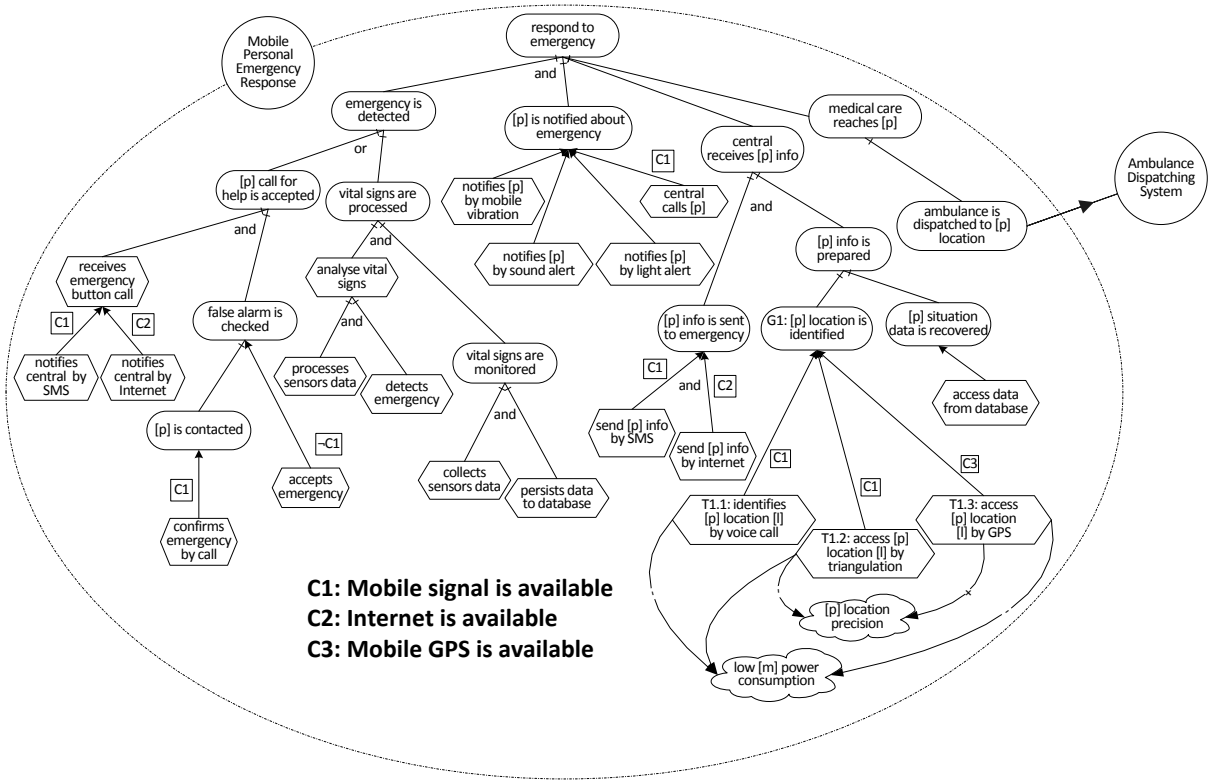

Figure 2: The Contextual Goal Model for MPERS

i.e., a system that will put patient's life at risk by sending a response team to the wrong location.

2.6 Related Work

Many state of the art contributions which involves Goal modeling for requirement analysis are present in the literature. Mylopoulos et al. [16] address the monitoring and diagnosis of requirements specified using goal models. Their framework is based on a runtime monitoring to identify and diagnose requirements violations. Our framework, differently, relies on domain knowledge to provide information about the system failures in different contexts of operations. Thus, it can be used to assess decision making of self-adaptation mechanisms without operational data, which can be considered a preventive plan to keep dependability.

Regarding the combination of goal modeling to fuzzy logic, Baresi et al. proposed adaptable Live goals that can be either crispy or fuzzy, i.e., whose satisfaction can be expressed by different degrees and not just true of false [4, 6, 5]. In contrast, our framework is based on crispy CDRs and the fuzziness is applied not to goals, but to the context facts whose intensity or quantity are better translated by fuzzy qualitative terms. In addition to that, fuzzy logic rules are used to create implications among context conditions and dependability levels of goals. Live goals allow the adaptation of the goal model itself. Our framework assesses the decision of which alternative solution to be used to fulfil system goals considering the actual context of operation and its impact on the dependability of different system tasks.

The Awareness Requirements (AwReq) defined by Souza et al. [15] can drive adaptation to maintain an acceptable achievement levels of goals through runtime system monitoring. In comparison to our approach, AwReq also provides criteria for self-adaptation and applies to dependability requirements. Our novelty is to provide means to the specification of the contextual dimension of requirements for dependability and to allow the specification of context impacts on goal dependability levels. AwReq use an iterative approach to define the success levels of requirements. It starts with high level qualitative words and ends with quantitative values. Their framework extrapolated the concept of softgoals by using measurable quality constraints (QC) in a goal model. Following this approach, dependability requirements are also defined as measurable QCs in the model. We would also add that these QCs could be context-dependent. This means that the required level of quality may vary according to the context of operation. This relation becomes more visible when the criticality of systems features is itself context-dependent. Thus, the degree of dependability of these features is also dependent on the context. Finally, another effort in this area is the specification language targeting self-adaptation which was proposed by Whittle et al. [17].

3. PROPOSAL

3.1 Model Description

The proposed approach is divided in three phases. First, the CDR is defined based on a context analysis and a failure classification that identifies, for each critical system goal and meaningful context of operation, the dependability attributes which are affected by contextual time and/or do-

main failures and their consequence for the system and environment if they occur. This information is used to define the contextual dependability requirements of different leaf goals in a goal model.

Second, the effect of contexts on the likelihood of failures (CFI) is modelled using fuzzy logic IF-THEN rules. Each rule addresses a single dependability attribute previously identified by failure classification for the CDR phase. These rules associate contexts conditions (IF) to the dependability of system tasks (THEN), as described in the diagram of Figure 3. Membership functions are required for the *fuzzification* of inputs (facts) and the *deffuzification* of outputs (dependability attributes).

Finally, the third phase defines two possible validation approaches for the existing CDRs. Firstly, static validation can be performed for all possible meaningful context variations formed by the space of context facts. This validation could indicate in which context conditions each goal would not be satisfied in terms of contextual dependability requirements. Secondly, a runtime adaptation loop mechanism could monitor the current context of operation and rely on CFIs to choose the most dependable solution among alternatives in the CGM.

3.2 Formalization

Our approach relies on the modelling and analysis of contextual failures in a CGM. We extend this conceptual model with new cause-effect relations. The class diagram in Figure 3 illustrates the extension which consists of a many-to-many association between contextual effect rules and the consequent level of dependability attributes of system tasks, i.e., the CFI. The many-to-one association between CDR and goals is also illustrated.

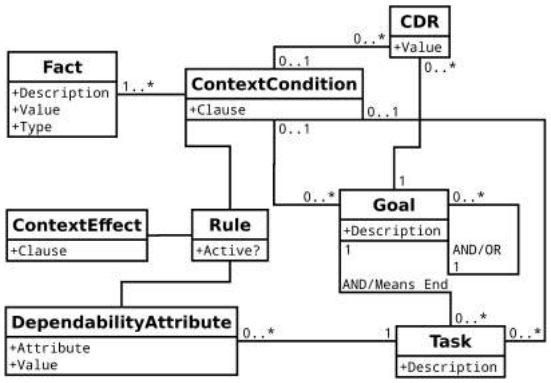

Figure 3: The metamodel of CGM Extension

Each task CFI is represented by a context condition, its effect in terms of magnitude and the dependability attributes affected. These last two are linked by a rule entity in the model. The CDRs are associated to zero or one context condition and linked to one specific goal. Goals are decomposed into tasks according to CGM's possible decompositions (AND, Means-End).

Fuzzy logic is also used to formalize the cause-effect relations between the fuzzy facts defining a context and the dependability attributes. Figure 4 illustrates the fuzzy inference with the elements of our approach.

Different rules may shape the effects of context on a single quality from different viewpoints. Each rule will be weighted by the membership degree of its input(s). If there are more than one fuzzy input for a rule, logical operators such as OR and AND will define the resulting membership degree to be used as a weight for this rule. Finally, all the active and weighed rules are aggregated following a predefined approach like the center of gravity (COG). The output is then calculated using the membership function for the defuzzification of the output [14].

The quality result is therefore based on the output of the fuzzy inference process. Deffuzification of the output allows a more precise comparison of the alternative ways to reach goals in terms of quality (dependability in our case) using numeric values. Nonetheless, the rules defined during the analysis phase are all based on the fuzzy or boolean levels of the facts describing a context.

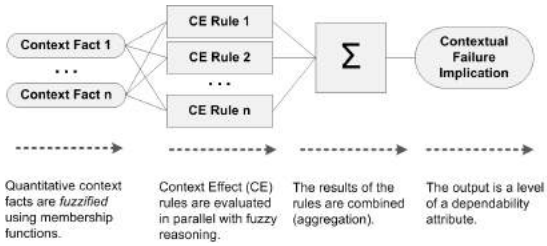

Figure 4: Fuzzy inference

Finally, the whole process of our approach, including both contextual dependability requirements elicitation and the contextual failure implication modelling, is illustrated by the diagram presented in Figure 5. We further elaborate on each activity of the proposal activities in their referred sections in Figure 5 as follows.

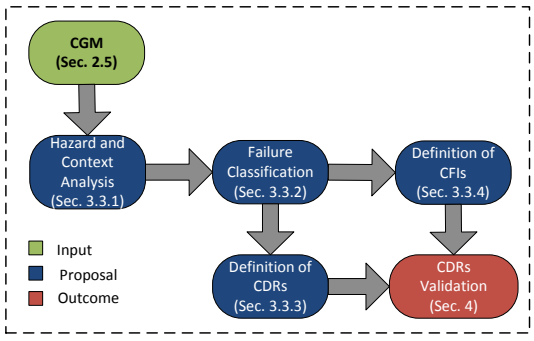

Figure 5: Proposal activities

3.3 Applicability

In this section, we use the MPERS system modelled via the CGM presented in Figure 2 to demonstrate the applicability of our approach. We exemplify the process of modelling and analysing contextual failures for an important feature of the MPERS system: the ability to identify a patient's location (G1) in order to send that patient an assistance if an emergency is detected. The tasks involved in this goal are T1.1, T1.2 and T1.3. These alternatives represent different ways of achieving G1: by voice call, triangulation or GPS.

3.3.1 Context Analysis

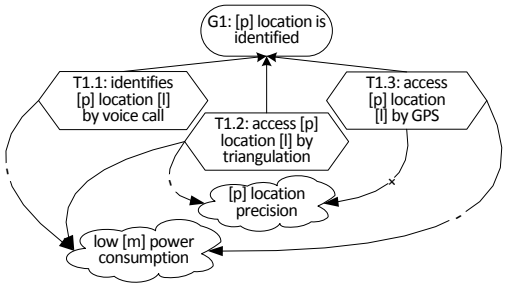

Figure 6: Identify patient's location goal and alternative tasks

Following the context analysis technique proposed in [1], context could be ultimately expressed as a formula of observable (monitorable) pieces of information, called Facts. In the original proposal, facts were treated as boolean variables and there was no consideration for facts representing variable levels of existence of a system resource or an environment state. In this paper, we consider not only the simple boolean facts, but also those associated with multiple levels of value. Moreover, we do not rely on crispy definitions for each of these levels, but rather on fuzzy ones that allow a flexible association of system resources and environment states to qualitative words such as low, average and high battery level.

By analysing the MPERS model from dependability perspective, a list of hazards were elicited. Their interpretation led to boolean and fuzzy facts that could potentially affect the behaviour of different tasks, i.e., they affect the likelihood and consequence of failures. Potential hazards and the contextual facts related to MPERS are listed in Tables 1 and 2, respectively.

Table 1: List of Potential Hazards

Potential Hazards
Mobile is out of battery
Sensors are out of battery
Central is not aware of emergency
Central is not aware of patient's location
Emergency is not detected through vital signs

Table 2: List of Facts which form context conditions

Fact	Type	Belongs	Description
1	Fuzzy	Mobile	Battery level
2	Fuzzy	Mobile	Voice/sms signal level
3	Boolean	Mobile	Data signal availability
4	Fuzzy	Mobile	Wi-Fi signal level
5	Fuzzy	Mobile	GPS signal level
6	Fuzzy	Mobile	Bluetooth signal level
7	Fuzzy	Mobile	Memory availability
8	Boolean	Mobile	Power source availability
9	Fuzzy	User	Health criticality
10	Fuzzy	Environment	Temperature
11	Fuzzy	Sensors	Battery level
12	Fuzzy	Sensors	Active amount

3.3.2 Failure Classification

To classify failures, each task failure related to meaningful hazards should be analysed in terms of the domain and consequence level. As stated before, the consequence level of failures may variate according to the context of operation (from minor to critical). Thus, the failure classification must associate this information to one or more contexts.

Following the consequence level definition, the corresponding dependability attributes affected by failures are identified based on both domain and consequence level. Table 3 presents the resulting classification for G1.

Table 3: Failure classification for goal G1

Domain	Consequence	Attribute
Time	If location data is not **fresh** then consequence is **minor**	availability
Time	If location data is **fresh** then consequence is **catastrophic**	availability, safety
Data	Always **catastrophic**	reliability, safety

Regarding failure classification, if , for example, the system requests a method which turns out to be unavailable, this will trigger a time failure. The consequence level of time failures depends on the freshness of last location data as the system should collect patient's location periodically. A data domain failure is related to the precision of the patient's location data delivered by each method. In case of imprecise or erratic data, emergence response team may be unable to send assistance to patient's location, therefore a data failure is related to the reliability of these methods.

In contrast, safety attribute is contextually related to both time and data failures. The unavailability of any method to identify a patient's location would have a catastrophic consequence only in the context where no fresh data has been collected before. On the other hand, the imprecision in location data is severe even if fresh data is available once it could lead the medical assistance to the wrong location (as if patient have moved since last received data). Accordingly, in this example safety attribute is only contextually related to time failures and is always related to data failures.

3.3.3 Contextual Dependability Requirements

Following the failure classification, the contextual dependability requirements for G1 were defined. The consequence of failures and other factors may influence on the required levels of dependability attributes for system goals. As most features in an emergency response system are critical, contextual dependability requirements may vary from high level to at least average level in certain contexts like 'location data is fresh'. In Table 4 both CDRs and softgoals related to this goal are listed.

3.3.4 Contextual Failure Implication

In a mobile system environment, the resources needed by each of the methods for identifying patient's location are not static. Mobile signal used for voice calls and SMS can be attenuated or not available in remote regions, affecting the reliability or even the availability of the tower triangulation method. GPS signal can be blocked by buildings walls and even the patient's ability to answer a voice call may not exist due to some debilitation caused by an emergency situation.

Table 4: Non-functional requirements for goal G1

Type	Description/Constraint
CDR	If location data is not **fresh** then availability must be >= average
CDR	If location data is **fresh** then availability must be high
CDR	Reliability must be high
Softgoal	Patient's location precision
Softgoal	Low mobile power consumption

In order to select the alternative with more chances of successfully achieving its goals, context-aware systems must count on the information about when it is more appropriate or valid to use each method. Table 5 and Table 6 present a list of IF-THEN rules describing context effects on the qualities of alternative tasks involved in identifying a patient's location.

Table 5: Set of rules defining context effects on the availability of alternative G1 tasks

Rule	Task	Context Effect
1	T1.1	If situation is not **emergency** then availability is **none**
2	T1.1	If situation is **emergency** then availability is **high**
3	T1.1	If mobile voice/sms signal is **weak** then availability is **low**
4	T1.1	If mobile voice/sms signal is not **weak** then availability is **average**
5	T1.2	If mobile voice/sms signal is **weak** then availability is **none**
6	T1.2	If mobile voice/sms signal is not **weak** then availability is **average**
7	T1.3	If GPS signal is not **strong** then availability is **low**
8	T1.3	If GPS signal is **strong** then availability is **high**

Following these analysis, different outcomes for the approach may take place. In the next section, we briefly discuss the *runtime alternative selection* and present a tool for static CDRs validation.

4. REASONING WITH CONTEXTUAL DEPENDABILITY

4.1 Runtime Alternative Selection

To illustrate, we consider two runtime scenarios with different context conditions as described bellow:

- **First Scenario**
  - situation is **emergency**
  - patient health is **stable**
  - mobile voice signal is **80%**
  - GPS signal level is **30%**
- **Second scenario**
  - situation is **emergency**
  - patient health is **critical**
  - mobile voice signal is **75%**
  - GPS signal level is **90%**

Table 6: Set of rules defining context effects on the reliability of alternative G1 tasks

Rule	Task	Context Effect
9	T1.1	If Patient's health is **critical** then reliability is **low**
10	T1.1	If Patient's health is not **critical** then reliability is **average**
11	T1.1	If Mobile voice/sms signal is **weak** then reliability is **average**
12	T1.1	If Mobile voice/sms signal is not **weak** then reliability is **high**
13	T1.2	If Mobile voice/sms signal is **weak** then reliability is **none**
14	T1.2	If Mobile voice/sms signal is **average** then reliability is **low**
15	T1.2	If Mobile voice/sms signal is **strong** then reliability is **average**
16	T1.3	If GPS signal is **weak** then reliability is **average**
17	T1.3	If GPS signal is not **weak** then reliability is **high**

We do not extend this example to the runtime acquisition of the patient health condition as stable or critical. This could be achieved through the definition of rules to determine the current health condition based on vital signs collected by sensors. The other two inputs are fuzzified using corresponding membership functions. Also, a open source fuzzy logic tool was used to reason the membership functions for each context fact and dependability attribute and also the IF-THEN rules used for implication [8]. This tool was able to process the fuzzy inference steps and produce information about the membership degree of each input, activated rules and the numeric output for each dependability attribute after *deffuzification*.

The resulting fuzzy input levels and their membership degrees for each considered scenario are described bellow. Here and throughout the remaining of this paper we use the notation *qualitative word(MD)* to describe the pair of qualitative word representing a fuzzy fact or a CDR and their corresponding membership degree (MD).

- **First Scenario**
  - patient health: **stable(1.0)**
  - mobile voice signal: **strong(1.0)**
  - GPS signal level: **weak(1.0)**
- **Second Scenario**
  - patient health: **critical(1.0)**
  - mobile voice signal: **weak(0.5)**, **strong(0.5)**
  - GPS signal level: **strong(1.0)**

In terms of availability, rules 2 and 4 were activated for task T1.1 (by voice call), rule 6 was activated for task T1.2 (by triangulation) and 7 for task T1.3 (by GPS). The second

context scenario also activated rules 2 and 4 for task T1.1. Only rule 8 was activated for task T1.3. For each analysed task/quality pair the aggregation of activated rules generates the output in terms of a fuzzy qualitative level.

In terms of reliability, rules 10 and 12 are activated for task T1.1, 15 for task T1.2 and 16 for task T1.3. The second context scenario activated the same rules, except for Task 1.3 that activated rule 17. The aggregation of activated rules produces numeric output values of each attribute following the definition of the membership functions for *deffuzification* of each attribute. In this example, numeric quality values ranges from 0 (none) to 10 (max) for all dependability attributes.

According to the automated fuzzy inference process, the resulting attributes for each task are summarized bellow:

- **First scenario:**
  - T1.1: **6.00** availability, **7.22** reliability
  - T1.2: **6.00** availability, **6.00** reliability
  - T1.3: **3.00** availability, **7.12** reliability
- **Second scenario:**
  - T1.1: **4.25** availability, **5.60** reliability
  - T1.2: **6.00** availability, **4.50** reliability
  - T1.3: **8.73** availability, **8.73** reliability

The predicted quality of the goal 'identify patient's location' depends on the quality of the selected alternative task. The decision on which alternative to use is therefore based on the dependability requirements associated to this goal and also to the softgoals affected by each task. This decision may follow different approaches for multi-objective optimization, for instance attending all the related CDR and then considering the softgoals.

Regarding the dependability attributes involved in this analysis, **reliability** is described by the continuity of correct service, e.g., the precision of the location data. Hence, this is the first and foremost dependability attribute to be considered as satisfaction criteria to the selection of the alternative method to identify patient's location at runtime.

According to Table 4, the CDRs activated by each context are:

- **First scenario:**
  - CDR: reliability must be >= **average(1.0)**
- **Second scenario:**
  - CDR: reliability must be **high(1.0)**

The membership function of each attribute defines the range of values accepted as low, average and high. As these are fuzzy and not crispy sets, the membership degree of the CDRs must also be defined. To keep the definition of these constraints in terms of high level qualitative words used by the analysts and domain experts, we assume here that the membership degree of the CDRs are 1.0 if not explicitly defined, meaning that a *high* reliability will be parsed as *high(1.0)*. The acquisition of a quantitative value comparable to the output of the CFI we make use of the membership function for *deffuzification* of each attribute.

We assume here the lower limit of the range of numeric values where the membership degree is 1.0. More granular definitions for the CDRs could be achieved using explicit definitions for the membership degree. The corresponding numeric CDRs and valid alternative tasks are:

- **First scenario:**
  - Numeric CDR: reliability must be >= **5.00**
  - Valid tasks: T1.1, T1.2 and T1.3
- **Second scenario:**
  - Numeric CDR: reliability must be >= **8.00**
  - Valid tasks: T1.3

In the first scenario, alternative T1.1 has a higher reliability and therefore should be used. In the second scenario, only alternative T1.3 is valid for identifying patient's location. This allows a proactive planning of the system configuration to be used at runtime through mechanisms of self-adaptation. Further description of this coupling between our proposal and mechanisms for self-adaptation is part of our future work.

4.2 Static CDRs Validation Tool

In contrast to reliability, availability is related to the readiness of correct service. In this example, readiness of each alternative method should be verifiable at runtime. Thus, its value defined by the implication of contexts in our approach (CFI) is more suitable to be validated against the CDR at early requirements analysis. This information could be used to verify the CGM against the CDRs for all contexts conditions stated in the rules. This pre-processing should be accomplished by an automated tool capable of parsing the related contexts in the rules and evaluating the CFIs for these contexts. Finally, this reasoning should compare the resulting dependability attribute levels for each context against the corresponding CDR for that same context condition.

To put that in a more precise form and show it utilizes our model, we define a formal Reasoning with Contextual Dependability presented in Algorithms 1, 2 and 3. Also, an online Java implementation of our tool is available for verification [1]. An open source Java library of fuzzy logic was used to accomplish the step of fuzzy inference in CFI reasoning.

The purpose of these algorithm is to identify contextual violations of dependability in terms of all or some of its attributes. For instance, the MPERS model could be validated for the availability of its leaf goals. Consequently, goals that violates the availability defined by CDRs should be identified with the corresponding violating contexts. This information would be of great value for the analysts and stakeholders as it may determine if the CGM should be re-factored before moving to next phases of system design and implementation. It also enables analysts to get a richer picture of an anomaly in the model from the dependability perspective by answering questions related to the goal affected and the context of use and the severity degree.

The logic behind these algorithms is simple. First, all the leaf goals of the CGM are identified and listed. Second, all valid and possible contexts formed by the combination

[1] `https://github.com/danilomendonca/CDRTool`

Algorithm 1: reasonCDRs(CGM cgm, List [] CDRs, List [] CFIs)

Input: CGM, list of context facts, list of CDRs, list of CFIs
Result: List of violating contexts with corresponding goals and violated CDRs

```
1  Goals leafGoals ← getLeafGoals(cgm);
2  List [] invalids = new List;
3  foreach Goal goal in leafGoals do
4      List [] goalCDRs ← getAssoficatedCDRs(goal, cdrs);
5      CFI goalCFI ← getAssociatedCFI(goal, cfis);
6      List [] contexts ← contextsInCFI(goalCFI);
7      foreach Context context in contexts do
8          if !verifyCDR(goal, context, goalCDRs, goalCFI)
           then
9              invalids.append(new Array[goal, context,
               cdr]);
10     end
11     return invalids
12 end
```

of facts and their variable levels are then listed. For each leaf goal, we iterate over contexts used as input by the rules of the corresponding CFI. For each of these contexts, the corresponding CDRs of dependability attributes are listed. Finally, these CDRs are validated against the corresponding attribute values obtained by applying the CFI in each goal's task using the same context of the CDRs. If one of the alternative tasks is valid for all the required dependability attributes, then this leaf goal is valid for that context of operation, i.e., at least one alternative task satisfies all activated CDRs in that context. Otherwise, the goal should be appended to the list of invalid goal-context pairs. Therefore, this algorithm validates all the leaf goals of the CGM to corresponding CDR for each valid context of operation associated to CFI rules.

In case no rule is activated or defined for some specific goal-context pair and still a corresponding CDR exists, then no validation is performed, as no information about the dependability attribute for that goal and context is available. In the same way, if no CDR is defined for a CFI context, then the goal is considered to be valid in that context condition.

Regarding the sort of tasks decomposition (AND, Means-End), the method *revaquiredAttributeLevel* should use two approaches for the aggregation of the attribute value of the related leaf goal being validated. That is, if underlying tasks are defined by Means-End Decomposition, any alternative is sufficient to achieve the goal, thus the dependability attribute level of the goal equals the maximum attribute value of the related tasks. If tasks are defined by AND Decomposition, then the resulting dependability attribute level equals the minimum attribute level verified for the tasks.

5. LIMITATIONS

Our approach to the modelling and analysis of contextual failures has two main limitations:

- The number of rules formalizing a context effect is proportional to the amount of contexts that are meaningful for the system and that are going to be part of the analysis. As we consider not only boolean facts, the possible facts combination forming context conditions may grow beyond the reasonable and available effort for requirements analysis. In order to reduce the amount of possible contexts conditions, a proper failure classification must identify system goals and features whose high criticality justifies the analysis cost.
- The quality of the inferences and domain information used to define both membership functions and IF-THEN rules is paramount to the effectiveness of the approach. Hence, and especially for critical systems, it is always a target to increase the precision of information about the consequence of failures to system users and environment and also to the likelihood of failures to occur given the set of context conditions. Imprecision should be mitigated with proper effort of analysts and domain experts. More validation and refinement are still needed to consolidate our approach in such terms.

Algorithm 2: verifyCDRs(Goal goal, List [] CDRs, CFI cfi)

Input: Goal, Context, list of goal's CDRs, goal's CFI
Result: True or False according to the validity of the goal for all attributes in CDRs

```
1  List [] contextCDRs ← getContextCDRs(context);
2  List [] dependabilityAttributes ← getAttributes(cdrs);
3  foreach DependabilityAttribute attribute in
   dependabilityAttributes do
4      boolean validTask ← false;
5      CDR cdr ← getAttributeCDR(attribute, cdrs);
6      List [] goalTasks ← goal.getTasks();
7      foreach Task task in goalTasks do
8          if getAttributeLevel(context, cfi, task, attribute)
           >= requiredAttributeLevel(context, cdr, goal,
           attribute) then
9              validTask ← true;
10             breakfor;
11     end
12     if !validTask then
13         return false;
14 end
15 return true;
```

Algorithm 3: getAttributeLevel(Context context, CFI cfi, Task task, DependabilityAttribute attribute)

Input: A Context, a goal's CFI, a goal's task, a dependability attribute
Result: Corresponding attribute level of the goal's task for some context

```
1  List [] rules ← getTaskRules(cfi, task, attribute);
2  List [] facts ← context.getFacts();
3  List [] attributeLevels ← callFuzzyInference(facts, rules,
   tasks, attribute);
4  if goal.getDecomposition() == "AND" then
5      return MAX(attributeLevels);
6  else
7      return MIN(attributeLevels);
8  end
```

6. CONCLUSION AND FUTURE WORK

In this work, we presented a framework for Contextual Dependability Requirements. We aligned concepts of dependability and failure classification to the requirements of a Contextual Goal Model (CGM). Moreover, our framework allowed the definition of contextual failure implications that can be checked against the contextual dependability requirements. We used a Mobile Personal Emergency Response System to demonstrate the feasibility of our approach. The proper elicitation of CDRs and CFIs allows a more dependable requirements analysis whose outcome may follow complementary paths. First, the verification for multiple contexts of operation can identify if dependable requirements are met or violated. This allows an early identification of risks before systems are designed and implemented. An algorithm demonstrating this outcome was proposed in Section 4. Second, run time self-adaptation can benefit of the information concerning the dependability of different alternative solutions given some context of operation and attributes. Thus, this approach has a significant value for the improvement of dependability of systems both at the time of requirements elicitation and also at runtime.

In our future work, we intend to explore further the relation of context to failures and their definition. Once operation data is available, context failure implications may be refined in order to improve its accuracy. Also, the use of temporal logic may improve the definition of contextual dependability requirements and lead to more sophisticated verification. Mechanisms of self-adaptation may be coupled to this approach and use dependability criteria for adaptation decisions. Also, in our current approach we are heavily reliant on the analysts to provide the specification. For complex and evolving systems, this might be a hard task which requires much resources and big teams which could be unavailable especially for small and medium enterprises. We will investigate approaches of Crowdsourcing [10] used in the context of obtaining software-related knowledge [2] to involve users and clients in contributing knowledge about failures and the context of use and how that context affects the meaning and consequence of a failure.

7. ACKNOWLEDGEMENT

The research is supported by an FP7 Marie Curie CIG grant (the SOCIAD project), CNPq grant number 482280/2012-3, under edital MCT/CNPq 14/2012, and Bournemouth University – Fusion Investment Fund (the BBB and VolaComp projects)

8. REFERENCES

[1] R. Ali, F. Dalpiaz, and P. Giorgini. A goal-based framework for contextual requirements modeling and analysis. *Requirements Engineering*, 15(4):439–458, July 2010.

[2] R. Ali, C. Solis, M. Salehie, I. Omoronyia, B. Nuseibeh, and W. Maalej. Social sensing: When users become monitors. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ESEC/FSE '11, pages 476–479, 2011.

[3] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on*, 1(1):11–33, 2004.

[4] L. Baresi and L. Pasquale. Adaptive Goals for Self-Adaptive Service Compositions. In *2010 IEEE International Conference on Web Services*, pages 353–360. IEEE, July 2010.

[5] L. Baresi and L. Pasquale. Live goals for adaptive service compositions. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems - SEAMS '10*, pages 114–123, 2010.

[6] L. Baresi, L. Pasquale, and P. Spoletini. Fuzzy Goals for Requirements-Driven Adaptation. In *2010 18th IEEE International Requirements Engineering Conference*, pages 125–134. IEEE, Sept. 2010.

[7] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[8] P. Cingolani and J. Alcala-Fdez. jfuzzylogic: a robust and flexible fuzzy-logic inference system language implementation. In *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*, pages 1–8, June 2012.

[9] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1):3–50, 1993.

[10] J. Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

[11] N. G. Leveson. *Safeware - system safety and computers: a guide to preventing accidents and losses caused by technology.* Addison-Wesley, 1995.

[12] X. Liu. Fuzzy requirements. *IEEE Potentials*, 17(2):24–26, 1998.

[13] K. Lorincz, D. Malan, T. Fulford-Jones, A. Nawoj, A. Clavel, V. Shnayder, G. Mainland, M. Welsh, and S. Moulton. Sensor Networks for Emergency Response: Challenges and Opportunities. *IEEE Pervasive Computing*, 3(4):16–23, Oct. 2004.

[14] MathWorks. Fuzzy Inference Process. `http://www.mathworks.com/help/fuzzy/fuzzy-inference-process.html`, 2014. [Online; accessed 09-January-2014].

[15] V. E. Silva Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos. Awareness requirements for adaptive systems. In *Proceeding of the 6th international symposium on Software engineering for adaptive and self-managing systems - SEAMS '11*, page 60, 2011.

[16] Y. Wang, S. McIlraith, Y. Yu, and J. Mylopoulos. Monitoring and diagnosing software requirements. *Automated Software Engineering*, 16(1):3–35, 2009.

[17] J. Whittle, P. Sawyer, N. Bencomo, B. H. Cheng, and J.-M. Bruel. RELAX: Incorporating Uncertainty into the Specification of Self-Adaptive Systems. In *2009 17th IEEE International Requirements Engineering Conference*, pages 79–88. IEEE, Aug. 2009.

[18] E. S.-K. Yu. Modelling strategic relationships for process reengineering. Jan. 1996.