Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

An Extended Goal-oriented Development Methodology with Contextual Dependability Analysis

Danilo F. Mendonça

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientadora
Prof. Dr.ª Genaína Nunes Rodrigues

Brasília
2015

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenadora: Prof.ª Dr.ª Alba Cristina Magalhaes Alves de Melo

Banca examinadora composta por:

Prof. Dr.ª Genaína Nunes Rodrigues (Orientadora) — CIC/UnB
Prof.ª Dr.ª Vander Alves — CIC/UnB
Prof. Dr. Luciano Baresi — Politecnico di Milano

CIP — Catalogação Internacional na Publicação

Mendonça, Danilo F..

An Extended Goal-oriented Development Methodology with Contextual Dependability Analysis / Danilo F. Mendonça. Brasília : UnB, 2015.
47 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2015.

1. LaTeX, 2. metodologia científica

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

An Extended Goal-oriented Development Methodology with Contextual Dependability Analysis

Danilo F. Mendonça

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof. Dr.ª Genaína Nunes Rodrigues (Orientadora)
CIC/UnB

Prof.ª Dr.ª Vander Alves
CIC/UnB

Prof. Dr. Luciano Baresi
Politecnico di Milano

Prof.ª Dr.ª Alba Cristina Magalhaes Alves de Melo
Coordenadora do Mestrado em Informática

Brasília, 30 de janeiro de 2015

Dedicatória

Agradecimentos

Resumo

A static and stable operation environment is not a reality for many systems nowadays. Context variations impose many threats to systems safety, including the activation of context specific failures. Goal-oriented software-development methodologies (SDM) adds the 'why' to system requirements, i.e., the intentionality behind system goals and the means to meet then. Contexts may affect what requirements are needed, which alternatives are available and the quality of these alternatives, including dependability attributes. In order to allow a formal and probabilistic analysis of systems affected by context variation and elicited with Goal-Oriented Requirements Engineering (GORE) approach, we have proposed an extension to the TROPOS goal-oriented methodology to include dependability constraints to goals and to provide a more precise and formal requirements verification by translating a contextual goal-model annotated with a behavioural regular expression into a PRISM probabilistic model to be checked against properties defined with the Probabilistic Computation Tree Logic (PCTL). We evaluated the proposal with a case study of a Mobile Personal Emergency Response System (MPERS).

Palavras-chave: LaTeX, metodologia científica

a

Abstract

Keywords: LaTeX, scientific method

Sumário

1 Introduction **1**

1.1 Problem Definition . . . 1
1.2 Proposed Solution . . . 2
1.3 Evaluation . . . 4
1.4 Contributions Summary . . . 4
1.5 Document Organization . . . 5

2 Baseline **6**

2.1 Goal-oriented Requirements Engineering . . . 6
2.2 TROPOS Goal-oriented Software Development Methodology . . . 7
2.3 Contexts . . . 8
2.4 Dependability Analysis . . . 9
2.5 PRISM Probabilistic Model Checker . . . 10
2.6 Mobile Personal Emergency Response System . . . 12
2.7 Antlr Language Recognition Tool . . . 13

3 Related Work **14**

3.1 Contextual Goal Model . . . 14
3.2 Awareness Requirements . . . 15
3.3 Runtime Goal Model . . . 15
3.4 Dependability Contextual Goal Model . . . 16
3.5 Formal TROPOS . . . 17

4 Modelling the Problem **19**

4.1 Motivation . . . 19
4.2 Requirements . . . 20

5 TROPOS Methodology with Probabilistic Requirements Verification **22**

5.1 TROPOS Requirements Engineering Phases . . . 22
5.1.1 TROPOS Early Requirements Phase . . . 22

5.1.2 TROPOS Late Requirements Phase 24
5.2 TROPOS Extended Verification Phase 25
5.2.1 Runtime goal model 25
5.2.2 RGM-UML activity diagram comparison 26
5.2.3 Non-functional requirements specification 28
5.2.4 Non-functional requirements verification 30
5.2.5 Probabilistic model generation 32
5.2.6 From NFR to PCTL properties 33
5.3 Treating NFR Violations 34
5.4 TROPOS to PMC Code Generation 35

Lista de Figuras

5.1 MPERS at TROPOS early requirements phase 23
5.2 MPERS at TROPOS early requirements phase 24
5.3 MPERS tasks represented by a UML activity diagram 27
5.4 MPERS non-functional requirements. 28

Lista de Tabelas

5.1 Description of RGM textual notation used by the proposal. 26
5.2 asds 29
5.3 Description of the different approaches for verifying a system with variable alternatives and variable contexts. 31

Capítulo 1

Introduction

1.1 Problem Definition

Among the different causes that lead a system to fail, some can be tracked back to design decisions in early system development process. The systematization provided by requirements engineering has been used to improve the quality of the delivered system documents, models and specification. Nonetheless, in many cases the RE process do not further investigate or verify how the system-to-be will perform based on its design model, relying only on later execution testing.

Non-functional dependability requirements are important metrics related to the correct system operation. Dependability defines the ability of delivering a service that can justifiably be trusted. Fail forecasting should provide a qualitative and/or a quantitative evaluation of the system behaviour in respect to fault occurrence or activation, while fault removal includes the verification, diagnosis and correction [AVIZIENIS]. For some systems, the late fault correction may be very expensive and, in the worse case, may happen after a catastrophic event [DEPENDABILITY].

In traditional Goal-oriented requirements engineering (GORE) methodologies [GORE CAC], contribution analysis is based on domain knowledge about the positive, neutral (implicit) or negative impact of a given system alternative to one or more system goals, generally a qualitative goal. By comparing the overall contribution of two or more alternatives, a decision is made about which one should be adopted for the system-to-be. For instance, if one goal is to communicate with a remote user mobile, alternative means for the notification agent could be to send a SMS, an internet based message or a voice call. These alternatives may contribute with different values for qualitative goals such as 'reliable delivery', 'fast delivery', 'convenient delivery', etc.

The problem with this approach is threefold. First, it is based on domain knowledge information that may not exist or may not be precise and reliable. As a consequence, the

decision of which alternative to use, either at design time or at runtime, may be biased and lead to unexpected violations - the selected alternative may be unable to fulfil its qualitative goals. Second, it is limited to a static representation without any dynamic information that could be used to verify quality attributes that depend on system behaviour and its many nuances such as execution order, cardinality and execution priority. Finally, contribution links are deterministic. As such, the design decision based on contribution analysis is reduced to a simple sum comparison of the concurring alternatives contributions with no support for probabilistic verification.

Additionally to the contribution analysis limitation, the context in which systems operate may not be static. Mobile and pervasive computing, among others, are examples of new computer paradigms for which the environment is dynamic. Battery, signals strength, components availability and the quality of physical resources and relevant information such as the user geographic location may vary through time, posing a new sort of challenge to the development of socio-technical systems based on these paradigms. The contextualization of the informations gathered at RE phase becomes imperative once its validity may be threatened by changing environment conditions [Finkelstein, CGM]. Accordingly, any improvement of the GORE contribution analysis may have to consider multiple operational contexts.

Static model-based verification is a powerful approach to evaluate metrics over a finite-state model representing the system behaviour. The advantage of this approach is to reduce the correction costs by anticipating failures at early stages of the development cycle. Increased analysis overhead is one of the drawbacks of this approach. Thus, its application must be well justified by the criticality level of the system and by the adoption of an appropriated model-based verification approach.

Despite its valuable contribution to the requirements analysis and engineering, GORE still lacks the proper means to verify the conformance of some important metrics such as dependability requirements and others non-functional requirements (NFR) related to the likelihood and frequency of system execution failures. By tackling the verification of these metrics with a more precise and reliable approach, this work aims to mitigate the occurrence of failures at an early phase of the system development that must justifiably be trusted.

1.2 Proposed Solution

In order to provide a more solid and precise approach for the non-functional verification of different system alternatives and to improve the GORE contribution analysis, we propose the extension of the TROPOS goal-oriented software development methodology

with a probabilistic model checking (PMC) approach that has already been explored and is supported by tools such as PRISM model checker[genaína PMC, PRISM]. PMC is a formal method for static verification of system models. System models should represent specified activities that fulfils the root goal or any lower level goal and its elicited alternatives. This model should then be checked for properties such as reliability, availability, performance and power consumption.

The PMC technique requires a behaviour system specification. As the goal model proposed in TROPOS is static, no information regarding achievement/execution order, cardinality and priority of goals/tasks is available, except the activity diagram for the detailing of an agent's single capability behaviour and the sequence diagram for agents interaction. Nonetheless, this problem was tackled by Dalpiaz et al. with a regular expression language to express runtime information, e.g., how many times the same goal should be achieved and the execution order of different system tasks [RGM].

We have used the RGM proposal to fill the gap between the static goal model and its dynamic representation. This dynamic view of the goal model is translated to a probabilistic model following the PMC technique and the PRISM model checker as the tool to automate the verification. PRISM language is used to define the probabilistic model and the Probabilistic Computation Tree Logic (PCTL) to describe the properties to be verified in the model. These properties are derived from the NFR associated to corresponding system goals being verified.

To address the problem of a dynamic context of operation, the context effects over goals, means and metrics should be parametrized to produce a formula that can check the system and its alternatives for different contexts. This verification, performed as part of the Validation & Verification (VV) phase in RE, should anticipate (contextual) violations of non-functional requirements.

Treating a detected violation at design time may correspond to actions such as making a different choice for underlying components used by this alternative's tasks, optimizing its behaviour specification or even the disposal of this alternative as a means to satisfy its goal if there is at least one other valid alternative. PMC technique also allows the identification of system alternatives with more influence on each metric through sensitive analysis.

By coupling a formal verification to a goal model, our approach benefit from a clearer understanding of the system-to-be and its criticality, justifying the verification of system parts or the system whole. The automatic generation of the verification model from an extended version of the goal mode, namely the runtime goal model, greatly reduces the verification overhead. Finally, including context effects provide a more realistic representation of the system to be verified.

Runtime self-adaptation is beyond the scope of this work. However, based on the contextual analysis provided by the CGM and the enriched non-functional and dependability analysis provided by the verification of different alternatives using the PMC technique, it should not be difficult to extend the approach with the additional monitoring, planing and execution capabilities of a self-adaptation loop and have a self-adaptive architecture and mechanism reflected upon its runtime goal model requirements. These concerns should be addressed in future work.

1.3 Evaluation

This proposal was evaluated with the application of the extended TROPOS methodology to the development of a Mobile Personal Emergency Response System (MPERS). This system may be seen as a body area network (BAN) with extended functionalities related to ubiquitous emergency response running in a mobile device [BAD]. Instead of a home or hospital static environment, the MPERS is conceived to allow patients with different health risk degrees to maintain mobility while they are monitored and assisted. If a medical emergency is detected, a geolocation feature should point out the location where the emergency response team must be addressed to.

The MPERS features were based on real emergency response systems available at the industry and also at the BAN explored in previews work [Fernandes].

The evaluation process was focused in revealing the major benefits and limitations of the extended TROPOS proposal. Time to market is an important aspect for any software development methodology. Also, the soundness and precision of the proposed probabilistic verification is crucial and must be evaluated as they should not result in mislead decisions about which alternatives should be used by the system. Instead, they must anticipate any violation that could lead to a system failure, specially severe or catastrophic failures, giving analysts valuable information about where the system requirements and specification should be tailored and improved.

1.4 Contributions Summary

This section summarizes the contributions of this proposal.

1. A new contribution analysis approach for the TROPOS Goal-oriented software development methodology.

2. Inclusion of context effects over goals, means and metrics in the probabilistic model using appropriate constructs and parameters for each case.

3. Conversion rules between different decomposition and runtime constraints in a runtime goal model to a probabilistic model in PRISM language.

4. A parser implementation for the regular expression (regex) language used in runtime goal models with support for execution order, cardinality, alternative execution, optional execution and conditional execution.

5. An automatic generation of the PRISM model representing activities from a runtime goal model annotated with the runtime regex and graphically modelled using the TAOM4E tool that supports TROPOS methodology.

1.5 Document Organization

This dissertation is organized as follows. Chapter 2 presents the base concepts of this work and the most important related works. Chapter **??** details the problem tackled by this proposal. Chapter 5 presents the new extended TROPOS methodology, the rules for the translation between the contextual goal model and the probabilistic verification model, the parser for the runtime regex and finally the implementation approach for the automatic generation of the probabilistic model in PRISM language. Chapter **??** evaluates the proposal and describes its benefits and limitations. Finally, Chapter **??** concludes this work with final considerations about the current proposal, related proposals and our future work.

Capítulo 2

Baseline

2.1 Goal-oriented Requirements Engineering

Goal-oriented requirements engineering brings forward the intentionality behind system requirements. More than just presenting the *what* and the *how* of a system-to-be, it provides the justification for each requirement, that is, they also present the *why*. Through a directed graph tree that begins with a root goal, goals are connected trough decomposition links. Root and higher level goals are related to strategical concerns, while lower level and leaf-goals are related to technical and operational features of the system.

The main purpose of a goal model is to support the early process of RE, including the elicitation of social needs and dependencies, the actors involved in delivering functionalities and resources, the decomposition of higher-level goals into more granular and detailed requirements chunks, the operationalization through means-end tasks and finally the comparison between different alternatives for the system-to-be. A goal model is said to be valid and complete if it follows all its syntactic rules and if all system goals are either decomposed, delegated to other actors or fulfilled by operational system tasks.

Three frameworks/methodologies, namely KAOS, i* and TROPOS, represent the foundations for the goal model analysis used by a variety of other proposals [KAOS, i*, TROPOS]. Despite some differences among their syntax, they all share a set of core concepts:

Entities

- **Actor:** an entity that has goals and can decide autonomously how to achieve them. They represent a physical, social or software agent. E.g.: A patient, an emergency center, a doctor and a Mobile Personal Emergency System running in patient's smartphone.

- **Goal:** are actors' strategic interests. A goal with a clear-cut criteria for its satisfaction is called a hard goal. In opposition, softgoals has no clear-cut criteria for deciding whether they are satisfied or not and are usually associated to non-functional requirements of an actor. E.g.: vital signs are monitored, emergency is detected, emergency center is notified (hard goals) and system availability, detection precision, emergency awareness (softgoals).

- **Task:** an operational means to satisfy actors' goals. E.g.: monitor temperature sensor, persist vital signs data, request emergency assistance.

Relations

- **AND/OR Decomposition:** a link that decomposes a goal/task into sub-goals/sub-tasks, meaning that all (at least one) decomposed goal(s)/task(s) must be fulfilled/executed in order to satisfy its parent entity.

- **Means-end:** a means to fulfil an actor's goal through the execution of an operational task by the same actor.

- **Contribution link:** a positive or negative contribution between a given goal/task to a softgoal. Contribution links are used for deciding between alternative goals/tasks at design time (contribution analysis).

2.2 TROPOS Goal-oriented Software Development Methodology

TROPOS is a GORE methodology based on the i* framework. Its main improvement is the addition of new phases of requirements engineering and system design, namely [TROPOS]:

- Late requirements engineering: Beyond the social dependency modelling with actors diagrams representing stakeholders and their needs in early

requirements phase, a late requirements phase focuses on the system actor analysis. In this phase, system goals are inherited from stakeholders needs and represent both functional and non-functional requirements. Each goal has to be further decomposed in more granular sub-goals, delegated to other actors or to be fulfilled by means-end tasks.

- Architectural design: In this phase, new actors representing sub-systems are created to fulfil different system goals. The idea is to shape the solution using a multi-agent architecture style instead of a monolithic system approach. Data and control interconnections are represented as dependencies.

- Detailed design: The last phase is characterized by the specification of agent capabilities and interactions though UML activity and sequence diagrams. Also, the implementation platform and other specific implementation details are addressed in order to directly map the design to system code.

Implementation phase is also specified by TROPOS methodology, but it is out of the scope of this work as our objective is to improve the analysis and the solution that will be later implemented.

2.3 Contexts

Context may be defined as the reification of the environment that surrounds the system operation [FINKElSTEIN]. Contexts, as already stated, may not be static, but dynamic. A system has no control over its context of operation. Accordingly, a system must be able to support different contexts of operation without violating its goals. Moreover, systems should be able to monitor the state of its surrounding environment and decide which alternative will be used regarding both the availability of that alternative and the optimization of non-functional requirements.

In GORE, dynamic contexts may affect what goals a system have to reach, the means available to meet them and also the quality achieved by each alternative[CGM]. Root goal and higher level strategical goals are not contextualized as they represent the main purpose of a system [Finkelstein]. As these goals are decomposed in more granular sub-goals, a context condition may dictate:

1. If the goal is required for that context, limiting 'what' a system should do;

2. If a sub-goal or task is adoptable, limiting the 'means' to fulfil a required goal;

3. The positive, neutral or negative contribution of using some goal or task to another goal, usually a qualitative softgoal;

The third effect is the main focus of this work, as it is related to the GORE contribution analysis that we aim to improve.

2.4 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 [AVIZIENIS]. According to Avizienis et al., dependability encompasses the following attributes:

- Availability: readiness for correct service.

- Reliability: continuity of correct service.

- Integrity: absence of improper system alterations.

- Safety: absence of catastrophic consequences on the user(s) and the environment.

- Maintainability: ability to undergo modifications and repairs.

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. The severity of failures consequence is related to safety.

An execution failure is a perceived deviation from system expected behaviour that may have variable degree of consequence on the user(s) and the environment. These failures are caused by specification faults or specification violations. In the first case, requirements model fails to describe the system and either the goals or the means to fulfil then are incorrect or incomplete. In the second case, software or hardware behaviour did not follow its specification due to a natural phenomena, a human-made, a malicious or an interaction fault [AVIZIENIS].

The scope of this work is restricted to specification violations, i.e., we assume that a system specification is complete and consistent. Failures are restricted to anomalous behaviour of the components participating in the execution of system tasks, including technical components and human actors. Regarding the different means to attain dependability, our proposal consists of a fault forecasting as part of the Validation & Verification phase of RE by estimating metrics related to dependability and assuring their conformance to the non-functional constraints associated to the goal model.

2.5 PRISM Probabilistic Model Checker

A model checking is a formal method that aims to automatically verify if a system model meets its specification for defined properties. Probabilistic and

state based model checking supports the verification of finite-state probabilistic models such as discrete-time Markov chain (DTMC), continuous-time Markov chain (CTMC) and Markov decision process (MDP), among others. Different types of properties may be defined to verify a system model for different qualitative metrics.

The PMC technique used in this approach is supported by the PRISM model checker tool [PRISM]. PRISM allows the modelling and analysis of systems which exhibit random or probabilistic behaviour. The decision of using PRISM as the probabilistic state based model checker was due to the number of successful case studies that have used this tool, indicating its maturity [PRISM CS], and also due to its rich environment that is able to represent different kinds of probabilistic models and their evaluations.

PRISM is suitable for many different kinds of model evaluations depending on the abstraction level, the type of probabilistic model and the PCTL properties to be analysed. PRISM language offers a rich set of constructs that may represent system modules and components, among others architectural and design configurations. Both qualitative and sensitive analysis are available.

As it will be explained in later sections, goal models may be easily extended with the proper information required for the verification of some important dependability attributes. The objective is to anticipate non-functional dependability violations and to support the decision of which alternatives to use in the system-to-be. A model checking technique should be used for dependability analysis as long as:

- A formal system model may be built;
- Properties representing dependability attributes may be defined;
- The analysis overhead is justified, e.g., by its criticality.

Finally, PRISM also supports a parametric model verification. That is, instead of providing the final evaluation for a given property, models may use parameters instead of initialized variables and the verification will output a parametric formula whose evaluation will check the model for any valid combination of parameters values.

2.6 Mobile Personal Emergency Response System

The MPERS case study will be further detailed in later sections as the goal models generated by TROPOS methodology are themselves useful for communication purposes. As such, this section will cover some non-functional aspects of this system that justify the use of our formal verification approach.

An emergency response system is a mission-critical system for which failures in achieving its main goals by the time they are required may lead to catastrophic consequences on users, i.e., on patients monitored by the system expecting to be promptly assisted in case of a medical emergency. Accordingly, any stakeholder that wishes to offer a service based on this system will have both ethical and contractual obligations regarding the safety of its product, that is, it must use appropriate means to prevent system failures.

Other dependability attributes such as reliability and availability are metrics over the correctness of system behaviour. MPERS is expected to have a high availability - as it must be ready to respond to an emergency that may happen at any time - and a high reliability - as an incorrect emergency response may lead to death or to costly false-positives. Integrity is a less critical attribute in this case, but must also be addressed as patient privacy may not be violated by disclosing his personal health or geolocation info to unauthorized persons. Maintainability is addressed by the use of a software development methodology and by the ability to update emergency rules remotely at runtime.

Reliability verification of an Ambient Assisted Living System also based on body-area networks though PCM technique was explored by Fernandes [Fernandes, 2012]. Reliability estimation demonstrates the non-determinism in the verification model that will result in an non-deterministic evaluation result. Moreover, PRISM cost/reward structure is illustrated by the power consumption verification. Analogue approaches could be used for other attributes with non-determinism and cost/reward structures in the model. The value obtained by this quantitative analysis of both attributes must comply with the corresponding non-functional constraint associated to the goal model.

2.7 Antlr Language Recognition Tool

ANTLR or Another Tool for Language Recognition is a open source parser generator for reading, processing, executing or translating structured text or binary files. The main purpose is to automatically generate a parser for a custom language defined in a specific grammar language supported by the tool. The parser can then be imported in any version compatible JAVA project to build and walk parsed trees.

As a result, any domain-specific language may be specified and then parsed using JAVA methods that will manipulate primitive attributes and objects according to what each parser rule and lexical term means for that language. In our proposal, ANTLR was successfully used to generate the parser for the regular expression language that specifies the behaviour of a runtime goal model (RGM). Further details of the grammar with both parser rules and lexical terms is given in later section.

Capítulo 3

Related Work

3.1 Contextual Goal Model

The Contextual Goal Model (CGM) [CGM] proposes the contextualization of required goals, adoptable means (goals/tasks) and contribution links values. The main benefit of this work is to enrich the original goal model with the contextualization of entities and relations affected by context variations and to provide a rationale for context analysis. In contrast, the main problem tackled by the our work is the verification of non-functional attributes such as dependability attributes that needs a more precise and less biased approach instead of the existing contribution analysis that is based on analysts direct evaluation of the forward impact between goals/tasks and softgoals.

In this regard, the CGM provided more realistic and precise contribution analysis contextualized by environment conditions, but did not change the nature of the contribution analysis process. Our work has benefited from the CGM conceptual model and has extended the non-functional GORE analysis with a context-dependent formal verification, i.e., that includes different context effects in the probabilistic model used by the PMC to contextually estimate the values of required non-functional attributes of the system and provide a reliable decision criteria for the selection of concurring alternatives in the goal model given different contexts.

3.2 Awareness Requirements

Souza et al. [AwaReq] proposed the Awareness Requirements (AwReq) as a meta-requirement in a goal model, i.e., AwReq specify the success/failure rate and temporal constraints for other requirements in the model, including goals, tasks, domain assumptions and other AwReqs (*-meta-requirement). The objective is to enrich the original goal model and provide constraints for system behaviour and to support self-adaptation, as runtime AwReq violations should be addressed by corrective actions. AwReq are formalized by a temporal logic formula, namely the Object Constraints Logic with Temporal Message (OCLtm).

Despite its contribution to the specification of meta-requirements in goal models, AwReq do not provide an approach to analyse and validate its meta-requirements before system implementation and monitoring. Original GORE contribution analysis could be used to define the impact of a given alternative to some attribute or value composing one or more AwReqs. However, the paper have only mentioned the formalization and monitoring through code instrumentation. In contrast, our approach relies on model based verification of meta-requirements similar to AwReq through probabilistic model checking technique that can be performed at design time and provide alternative design decision criteria and anticipate violations that must be treated before implementation.

3.3 Runtime Goal Model

Despite the use of goal models to support the runtime monitoring and adaptation, Dalpiaz et al. argued that these works are 'using design artefacts for purposes they are not meant to, i.e., for reasoning about runtime system behaviour'. As such, they proposed a conceptual distinction between the static goal model, named Design Goal Models (DGM), and the Runtime

Goal Model (RGM) that extend DGM with 'additional state, behavioural and historical information about the fulfilment of goals' [RGM].

The main purpose of the RGM approach is to provide the proper specification of behaviour information among system goals. RGM defines a class model, while the Instance Goal Model (IGM) captures instance states of runtime monitored goals that must conform to its class specification. IGM are useful to have an instance representation of the RGM provided by the monitoring of the activities involved in fulfilling system goals. If the monitored IGM violates the RGM, then a corrective action should take place. Our work has benefited from the RGM specification language by using the proposed regular expression to have a behaviour specification and generate the probabilistic model. Instance representation is out of the scope of our proposal.

3.4 Dependability Contextual Goal Model

The current work has been preceded by another proposal concerning goal-oriented requirements engineering, dependability analysis and dynamic contexts, namely the Dependability Contextual Goal Model (DCGM) [DCGM]. The contribution was focused on both dependability requirements and estimations based on declarative fuzzy logic rules and a variable context of operation.

In DCGM, a failure classification scheme was used to classify the consequence level and domain of failures in achieving system goals. This process lead to the definition of dependability constraints that must be achieved by the means-end tasks used to fulfil leaf-goals in specific contexts of operation, i.e., to the specification of contextual dependability requirements. These requirements inherited the same concept of the AwReq, but instead of being static, they could be associated to a context condition. Another facet of the DCGM is the contextual failure implication, which consisted of a dependabi-

lity specific GORE contribution analysis supported by fuzzy logic to define IF-THEN rules between context conditions and the level of a dependability attribute, e.g., availability and reliability.

The main drawback of this proposal was the lack of scalability, as declarative rules must be provided for different goals, attributes and contexts, proving to be a time-consuming task for the analysts. A second problem was the subjectivity of the rules, as they were based in domain knowledge that was also used to shape membership functions. This problem, as much as in GORE contribution analysis, lead to the idea of coupling a more precise and reliable verification approach such as the PMC technique. Still, the idea of a failure classification and the specification of contextual non-functional requirements were kept.

3.5 Formal TROPOS

The idea behind the formalization of a goal model, as proposed by Formal TROPOS [FTROPOS], is to provide a verifiable specification of sufficient and necessary conditions to create and achieve intentional elements like goals, tasks and dependencies in the model and invariants for each of these elements. In addition to this, new *prior-to* links describe the temporal order of intentional elements. Also, cardinality constraints may be added to any link in the model. Finally, Formal TROPOS uses a first-order linear-time temporal logic as a specification language.

The nature of the verification proposed by Formal TROPOS is different from the PMC used by our work. Formal TROPOS aims to provide the information required for a consistency verification of the goal model. The verification is not only for the abstract TROPOS syntax of intentional elements and relations, but also to domain specific information of how each element is created and fulfilled in time. Once the model starts to have more elements

and relations, its consistency checking becomes non-trivial, justifying the use of a formal specification that can be verified by a model checker tool.

In contrast to Formal TROPOS, our proposal uses a probabilistic model checking technique for the verification of properties that depend on how activities in the model are organized in terms of time, cardinality, combination, non-determinism and how each activity individually contributes to the property being evaluated. For instance, if power consumption is to be checked, each activity has to be associated to a power consumption unit and the global consumption value is evaluated considering any non-determinism specified in the execution workflow. Thus, even if the Formal TROPOS language also provides behaviour specification in a goal model, it is tailored for consistency checking of requirements and not for the probabilistic verification of dependability and other non-functional requirements.

Capítulo 4

Modelling the Problem

4.1 Motivation

Goal-oriented requirements engineering (GORE) has gained the attention of both academic and industrial practitioners due to its ability to systematically model the intentionality behind system requirements. More than just presenting the 'what' and the 'how', goal models also express the 'why' of different requirements to exist. Its simple graphical notation allows non-technical stakeholders to take part in the analysis process and have a clear view of the system-to-be. Finally, automated model verification should avoid violations of the requirements specification.

TROPOS is a GORE methodology that also includes architectural and detailed design phases for the development of socio-technical systems. 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.

In TROPOS, as in other GORE frameworks, there is no coupling to any specific verification approach for dependability attributes and other non-functional requirements. Contribution analysis is used for the comparison and selection of alternative design solutions based on how each alternative

contribute to one or more system goals, usually qualitative softgoals. This approach, however, is not tailored for metrics depending on system behaviour, i.e., on the dynamic coordination of activities and components interactions.

Probabilistic fault-forecasting aims to derive probabilistic estimates about measures related to the behaviour of a system in the presence of faults. Dependability benchmark enables the characterization of the dependability and security of a system and the comparison of alternative solutions according to one or several attributes [AVIZIENIS 37]. This benchmark may be achieved by a probabilistic model checking technique coupled to the later phases of the TROPOS methodology, enabling the dependability benchmark of a goal model.

An important success factor for any software development methodology is a reduced overhead to the development effort, including domain knowledge and tools to support its process. PMC may significantly reduce the occurrence of system failures, but it also requires additional knowledge about its modelling language and verification steps. A goal model extended by behaviour specification overlaps with the UML activity diagram used by PMC. Moreover, an automatic model generation for the PMC is desirable to reduce the know-how and effort for the verification of metrics for the system, justifying the implementation of this generator as an extension of a TROPOS modelling tool.

4.2 Requirements

Based on the identified gap of a formal verification of non-functional requirements in a goal-oriented requirements engineering approach, we defined the following requirements that must be addressed by our proposal:

R.1 **Backward compatibility:** Extended TROPOS runtime regex and contextual notation added to the goal model must not modify the existing syntax and semantic of the original TROPOS methodology.

R.2 **Optionality:** The use of the probabilistic model checking as a formal verification approach of runtime goal models as an extended TROPOS methodology should be optional and not mandatory.

R.3 **Model generation:** The probabilistic model representing the activities of runtime goal models should be automatically generated.

R.4 **Tool integration:** The same development environment tool used for TROPOS modelling and analysis activities should be extended with the runtime regex, context notations and verification model generation.

R.5 **Static syntax support:** The verification model should be coherent to the static goal model syntax for the AND/OR decomposition of goals/-tasks and for the goal-task means-end relation.

R.6 **Dynamic syntax support:** The verification model should be coherent to the runtime goal model regex including tasks sequential or interleaved execution order, cardinality as well as alternative, optional and conditional tasks execution.

R.7 **Contextual syntax support:** The verification model should be coherent to the context effects over the activation of goals, the adoptability of sub-goals/tasks and over the individual quality metric of components.

Capítulo 5

TROPOS Methodology with Probabilistic Requirements Verification

This chapter will describe our proposal and apply it to the MPERS case study. Both TROPOS phases regarding requirements analysis will be fully presented, as long as the formal verification process that requires a behaviour specification and the additional contextual notation regarding context effects. Further details about the TROPOS methodology may be found in the reference literature [TROPOS].

5.1 TROPOS Requirements Engineering Phases

5.1.1 TROPOS Early Requirements Phase

In Early Requirements phase, stakeholders are modelled as actors and their needs as goals. Each actor may be the depender or the dependee of a goal, task or a physical or information resource. Only the main system actor and the application domain stakeholders are analysed, leaving sub-systems and detailed agents analysis to later development phases.

The MPERS sytem and its social dependencies are presented by the actor model in Figure 5.1. System actors and social actors are displayed in different colors. Among the stakeholders, the Emergency Center represents a private or public organization that will provide the the emergency assistance

service and product to patients. Patient and doctor respectively represent the assisted person and the medical responsible for defining and evolving the emergency detection rules as part of an evolutionary approach for personal emergency response. Finally, sensors retailer should provide the vital signs sensors required for monitoring.

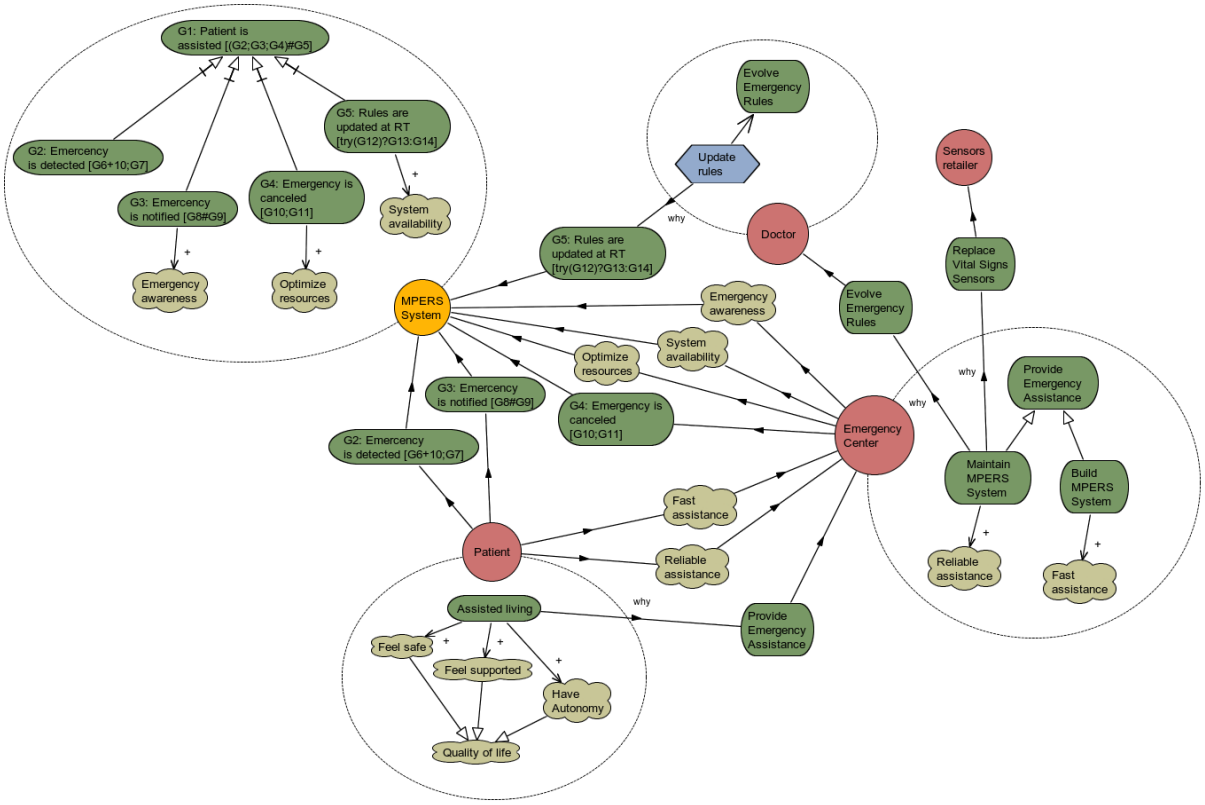

Figura 5.1: MPERS at TROPOS early requirements phase

From the diagram in Figure 5.1 it is possible to have a first view of the MPERS system-to-be. Main goals are divided in detecting, notifying and checking an emergency. Also, the ability to update the emergency rules at runtime (RT) is the fourth and last mandatory goal (AND-decomposition) that fulfils the root 'Patient is assisted' root goal. Functional system goals are strictly related to stakeholders functional and non-functional needs.

The yellow circle indicates that MPERS is a system actor. MPERS goals are already notated with a regex indicating its dynamic behaviour as part of the runtime goal model specification required by the proposal. This notation

is a reflex of the late requirements phase, as the TAOM4E tool supporting TROPOS methodology shares unique entities and relations among different development phases. The regex syntax is enclosed by brackets to differentiate then from goal name. In future work, a specific modelling compartment should receive the values for the runtime regex.

5.1.2 TROPOS Late Requirements Phase

Later requirements phase concentrates the analysis in the system-to-be and its operational environment. The MPERS goal model occupies the most part of the diagram and each of its main goals are further decomposed th-rough AND/OR decomposition. Also, means-end tasks defines how leaf-goals are fulfilled and the runtime regex across goals and tasks specifies dynamic properties of the system-to-be behaviour. Figure 5.2 illustrates the late requirements for the MPERS.

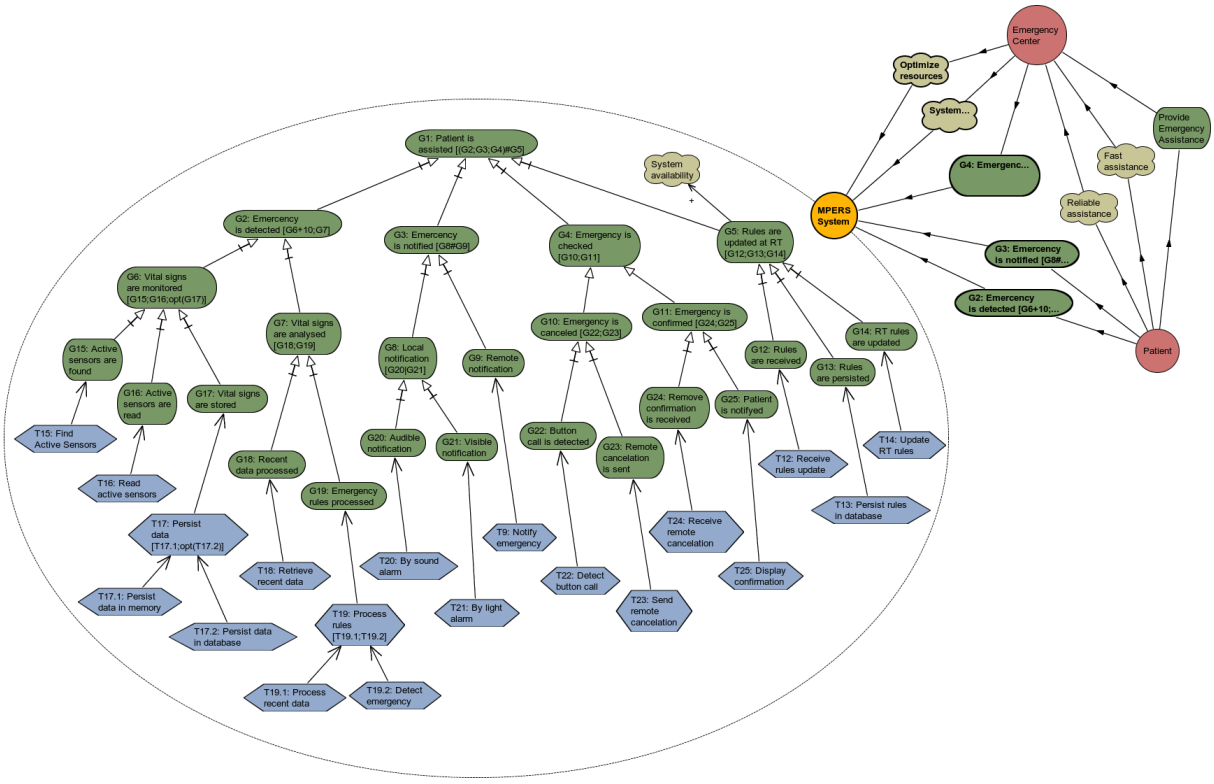

Figura 5.2: MPERS at TROPOS early requirements phase

At this point of the methodology, the system is represented as a monolithic actor and its goal model may be extended with the runtime specification. This extended model merges multiple views in the same diagram: strategical goals and operational tasks represent the requirements view for the system-to-be as well as the intentionality behind then, while the runtime specification provides a dynamic representation for the execution of operational tasks.

In our evaluation, we have first explored the use of the PMC technique for the verification of dependability attributes in the system model of the late requirements phase. The idea is to initially evaluate the approach in a monolithic representation of the system without the additional complexity of a multi-agent architecture. The evaluation involving later TROPOS phases should take place in future work. The remaining of this section will focus on the extended verification phase proposed by this work.

5.2 TROPOS Extended Verification Phase

To evaluate our verification approach with the MPERS case study, we have used a discrete-time Markov chain (DTMC) probabilistic model and focused on the verification of NFR related to dependability, i.e., NFR that are either direct attributes encompassed by dependability or that are related to one or more of these attributes.

5.2.1 Runtime goal model

Figure 5.2 already presents the runtime regex associated to system goals and tasks. The reference literature for the RGM was presented in Chapter 3. In this section, further details about the runtime regex syntax and semantic will be explained as they are a central part of this proposal.

Table **??** provides a textual description of each RGM notation with corresponding meaning in terms of what behaviour it specifies and also an example

from the MPERS RGM. The formal and detailed description can be found in the reference publication [RGM].

Expression	**Meaning**	**Example (MPERS)**
skip	No action. Useful for conditional ternary expressions involving two elements.	try(G10)?skip:G11
E1;E2	A goal/task E1 must be fulfilled/executed before E2.	G1;G2;G3
E1|E2	Fulfillment/execution of goal/task E1 is alternative exclusive with respect to E2.	T9.1|T9.2
opt(E)	Fulfillment/execution of goal/task E is not mandatory.	opt(T17.2)
E+n	Goal/task E must be fulfilled/executed n times, with n >0.	G22+2
try(E)?E1:E2	If goal/task E succeeds, E1 must be fulfilled/executed; otherwise, E2.	try(G10)?skip:G11
E1#E2	Interleaved fulfillment/execution of goal/task E1 and E2.	G8#G9
E#n	Interleaved fulfillment/execution of n instances of E, with n >0.	-

Tabela 5.1: Description of RGM textual notation used by the proposal.

5.2.2 RGM-UML activity diagram comparison

A similar specification could be provided by an UML activity diagram with activities as leaf-tasks of the goal model. However, activity diagrams have an homogeneous granularity level and do not clearly correlate behaviour to the requirements they are meant to satisfy. In contrast go the RGM, activity diagrams denote behaviour through graphical symbols, while the RGM mixes the original goal model notation with a textual runtime regex. This simple notation increases the utility of a goal model diagram. Figure 5.3 presents an activity diagram corresponding to the MPERS RGM.

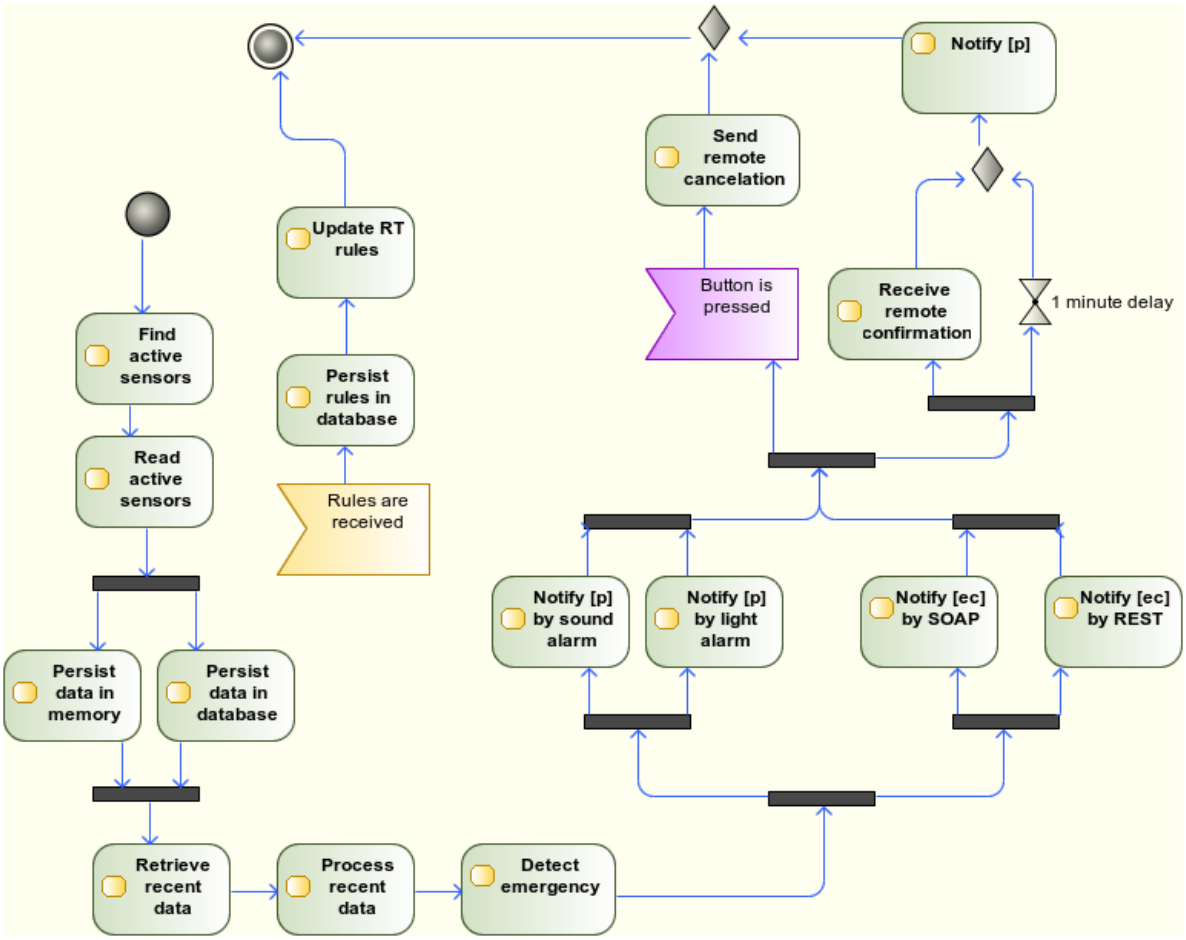

Figura 5.3: MPERS tasks represented by a UML activity diagram

Among its limitations, RGM does not express that an emergency has to be confirmed after a time or event condition is met as the UML activity diagram is able to express. Both necessary and sufficient conditions for the triggering and fulfilment of goals and tasks are provided by Formal TROPOS language. Still, sequential, parallel, alternative, optional and conditional execution flows as long as multiple executions of the same task can be expressed by the RGM, providing a rich behaviour specification that could be checked for meaningful non-functional requirements such as dependability attributes.

The idea of a runtime goal model is not to replace the activity diagrams, but to empower the goal model with a clear runtime syntax that could be used for documentation and communication and for conformance verification at both design time and during system execution (as it was originally proposed).

Depending on the complexity of the behaviour specification, a more robust runtime syntax would have to be employed or a standard UML behaviour diagram such as activity and sequence diagrams would have to complement the goal model.

5.2.3 Non-functional requirements specification

The TROPOS goal model also provides rationale for NFR analysis either via softgoals or qualitative hard goals. Figure 5.4 presents NFR for the MPERS system actor:

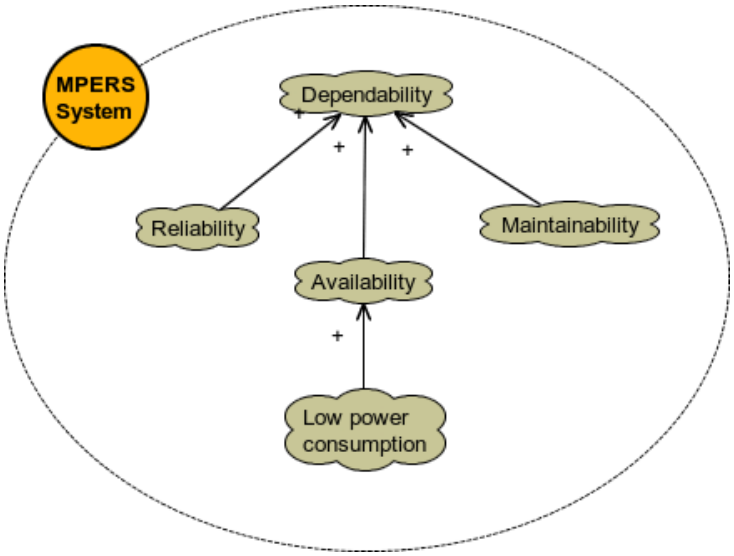

Figura 5.4: MPERS non-functional requirements.

Some dependability attributes, similarly to the Awareness Requirements by Souza et al. [AWREQ], define metrics over other requirements. These meta-requirements are not directly fulfilled by system functionalities like 'emergency awareness' is fulfilled by 'detect and notify emergency', but by how these functionalities will perform. Reliability, for instance, is inversely proportional to the likelihood of failures. Thus, the reliability metric depends on the probability of system functionalities to fail. Conformity to these metrics is not implicit in the model, it requires some verification technique.

Each requirement in a goal model must come from another requirement through decomposition, means-end or contribution links, or it must be directly

mapped to a stakeholder need through dependency links. Emergency center attended the patient's needs by providing and maintaining the MPERS system itself and by assuring other NFR for the system. Availability and reliability were selected as metrics over the system execution (meta-requirement), while maintainability is partially satisfied by the MPERS ability to update emergency rules at runtime and by other aspects that, for the sake of simplicity, we will omit in this evaluation.

It is no easy task to define the exact constraint to these metrics. First, an analyst or a reliability engineer should be aware of what does it mean to require, e.g., a 99.999% of reliability of a system. Second, this level may not be achieved by any of the alternatives available. In some cases, the system will have to comply to some standardized metrics. Table 5.2 summarizes each quantitative NFR for MPERS.

NFR	**Constraint**	**Target**
Reliability	99.9%	**G1**
Power consumption	100 p.u.	**G2**

Tabela 5.2: asds

As indicated by the *Target* column, each NFR constraint may be associated to a root level goal or to any of its subgoals. The corresponding PMC verification based on the execution of a set of system tasks is defined as:

- *Global*, if the activities set is a minimum set composed of the tasks that satisfies the chain of subgoals up to the root goal Groot, in this case identified as G1.

- *Local*, if the activities set is a minimum set composed of tasks that satisfies the chain of subgoals up to a goal Gx, where Gx != Groot.

Other MPERS non-functional requirements are 'emergency awareness' and 'resources optimization'. These are softgoals addressed by system functi-

onalities. The former receives a full contribution (double positive sign) from goal 'emergency is notified', meaning it is fully satisfied by this goal. The later is just assumed to be partially satisfied by the 'emergency is checked' functional goal.

5.2.4 Non-functional requirements verification

Given the possibility of an OR-decomposition in a goal model, more than one alternative can exist in terms of which subgoal should be achieved to satisfy its upper goal or which subtask should be performed to satisfy its upper task. Accordingly, multiple paths may lead to the satisfaction of the root goal. They will be called alternatives.

The probabilistic verification of systems with variability is not a novelty by itself. Many proposals have addressed this problem in the context of Dynamic Software Product Lines (DSPL), some of then using the PMC technique [Vini, Paula, Who Else?]. In DSPL, optional and alternative features may be activated or deactivated at runtime. A family based verification of one or multiple NFRs, also called qualitative goals, indicates what combinations are valid.

This variability leads to more than one minimum set of tasks capable of fulfilling local or root goals. Thus, the analyst should decide which alternative will be verified by passing parameters values (deterministic alternative selection, or DAS). DAS is useful for comparing system alternatives to decide which one should be used at design time or at runtime based on the evaluation of selected non-functional metrics. At design time, this approach is analogue to the TROPOS contribution analysis.

As in the CGM, contexts may limit which alternatives are adoptable. This effect must be considered in a realistic verification. For this, analysts may decide for which contexts NFR metrics will be verified (deterministic context selection, or DCS), or contexts may be selected following some probabilistic distribution (probabilistic context selection, or PCS). Both approaches are

complementary as the first verifies the selected alternative for one context at a time and the last verifies a real scenario representation with multiple contexts. Table 5.3 summarizes each approach and possible combinations.

		DAS	**PAS**
		A single alternative is selected by the analyst.	Alternative selection follows a probabilistic distribution.
DCS	An unique context is selected by the analyst.	Alternative selection by the analyst is limited by the selected context.	Probabilistic alternative selection is limited by the selected context.
PCS	Context selection follows a probabilistic distribution.	Alternative selection by the analyst may fail according to the probability of selecting an incomplatible context.	Probabilistic alternative selection may fail according to the probability of selecting an incomplatible context.

Tabela 5.3: Description of the different approaches for verifying a system with variable alternatives and variable contexts.

If the context selection is deterministic, there is no reason for verifying an alternative that is known to be incompatible with the selected context. In opposition, if context selection is probabilistic, that alternative may still be valid in other contexts. For instance, the alternative for identifying the patient location through GPS (alternative) will certainly fail if the GPS signal is not available (context). Thus, the DAS-PCS combination leads to the reliability verification of one system alternative for multiple contexts.

The idea behind a probabilistic context selection is to emulate a real scenario in which the context of operation varies and the system must avoid requirements violations by using a valid or optimal alternative. This holistic evaluation provides metrics that may be compared to non-functional constraints. If more than one alternative is compatible for a given context, prioritization should point out which one will be used.

5.2.5 Probabilistic model generation

In the PMC technique that has been adapted by this proposal, a behavioural specification, usually provided by UML activity and sequence diagrams, are manually converted to a probabilistic model in PRISM language. This tends to be a costly and error-prone process with a complexity proportional to the number of components, actions and interactions causing state transitions.

As a goal model is traversed from strategical root goal to operational leaf-goals, and each leaf-goal is reachable by a delegation to other actor or by a operational task, then a behaviour specification as proposed by the RGM may have enough information to be consumed as input for the generation of probabilistic models in PRISM language and for the verification of some important NFRs. However, the manual generation from RGM is still a costly task.

Depending on the abstraction level and the nature of the verification, PRISM models may either very complex or may follow a clear pattern. For instance, PRISM modules can be used to represent leaf-tasks of a runtime goal model. Considering a DTMC model, a task workflow can be modelled as a sequence of probabilistic state transitions according to the behavioural specification parsed from the RGM. This probabilistic model follows a pattern that motivated the implementation of an automatic generation of DTMC models representing leaf-tasks execution directly from a runtime goal model.

Leaf-tasks are not necessarily atomic system tasks. The idea is to leave to analysts the decision concerning the abstraction level represented by the tasks that will be verified. For instance, the abstract task 'find active sensors' could be decomposed in more granular and concrete tasks related to the platform, architecture and language used for implementation.

The more abstract a task is, the more difficult is to obtain their individual metrics, as the trace between an abstract and the concrete system operation becomes less evident. Any NFR verification by the PMC technique requires

further qualitative information about individual parts involved in an activity, e.g., the reliability, performance, power consumption, cost and other attributes for tasks in a workflow or for components involved in a task.

This is a key point in this approach, as it may seen too loose to couple a probabilistic verification to a goal model with a high-level operational representation of a system. But its feasibility becomes clear when the metrics being verified are compatible with the abstraction level of the tasks and information regarding how each task individually performs is available or can be collected.

5.2.6 From NFR to PCTL properties

The estimation of attributes through PMC technique is limited to those that a probabilistic model may evaluate. Dependability attributes have an abstract definition that must be associated to a concrete and verifiable PCTL property. To demonstrate our approach, we verify the MPERS model for the following attributes:

- **Reliability**, represented by the probability of a successful execution of all the activities involved in fulfilling leaf-goals of a certain system alternative. It is also know as the *reachability* as the describes the probability of reaching a final and successful system state.

- **Availability**, represented by the power consumption estimation to maximize the time that the system will remain operational depending only on its battery. This attribute is well related to mobile computing.

each task has its own states, including the failure and the success states. Many factors may contribute to the correctness or the failure of system tasks, including internal and external events. The probability of a successful task execution defines its reliability.

In a complex workflow of tasks with different rel

In order to be successful, tasks depend on the proper interaction among the components

seen as an activity diagram and be used to generate a probabilistic model in PRISM language. This allows the model checking of the corresponding goal model as a set of activities for which temporal and other behaviour aspects are specified by the runtime regex of the RGM.

5.3 Treating NFR Violations

- Making a different choice for underlying components: In some cases the replacement of a technical component for another of the same class can improve the quality of how they achieve their goal. For instance,

- Behaviour optimization: The quality may also depend on the pattern used for the activities execution. The specification of a different pattern may eliminate the non-functional violation.

- Contextualizing the alternative: An alternative may only violate a NFR in specific contexts. In this case, different valid alternatives may be used according to the context of operation.

- Alternative disposal: If the alternative is in absolute violation or if its validity is restricted to contexts that have at least one other valid alternative, this branch can be eliminated from the model.

Dependability analysis is used to provide information about different dependability attributes related to system failures. These metrics may be specified as non-functional requirements for isolated system functionalities or for the whole system. Instead of softgoals, we use meta-requirements over functional goals with clear-cut quantitative criteria such as '99.999%' reliable - a probabilistic value to make it compatible with the PMC estimation results.

To perform the , we focus on dependability related metrics that should be estimated and compared to their required constraint values through quantitative analysis. Sensitive analysis to reveal how different system parts contribute to the overall value of those attributes. Sensitive analysis may be considered analogous to the original GORE contribution analysis.

5.4 TROPOS to PMC Code Generation

As we wanted to automate the code generating process for the verification model, the graphical modelling environment that supports TROPOS methodology and the code generation for multi-agents was extended to also generate probabilistic models for the PMC technique.

To reduce the effort of codifying the verification model, an automated generation of the PRISM probabilistic model was implemented based on an existing open source tool for TROPOS development support named TAOM4E[citation]. TAOM4E provides a graphical environment for goal modelling with TROPOS methodology based on the well known Eclipse Modelling Framework (EMF) and Graphical Editing Framework (GEF). The GORE to PRISM generator was implemented as a Eclipse plugin and integrated to the TAOM4E environment.

The purpose of the automated code generation for the probabilistic PRISM model is to optimize the formal verification step by abstracting the PRISM language from the analysts and reduce the overhead and time of the model verification. This should increase the feasibility of adopting the extended TROPOS methodology by keeping analysts with their original responsibility of modelling and analysing the system, its social environment and its different contexts of operation.

In terms of a high level system behaviour, each activity has its own states space including success and failure. Our probabilistic verification approach requires not only a formal specification of the system behaviour, but also

metrics related to how individual components involved in system activities will perform in respect to the analysed metric. In reliability verification, each component has an individual probability of successfully performing its functional task. Analysts must obtain these values by consulting their manufacturer, by individually analysing each component reliability based on their behaviour specification until the atomic level or by monitoring these components in a testing or production environment. Further details of how individual metrics may be obtained for the PCM may be found at the literature and are out of the scope of this work.