

Pragmatic Requirements for Adaptive Systems: a Goal-Driven Modeling and Analysis Approach

Felipe Pontes Guimaraes^{1,4}, Genaina Nunes Rodrigues², Daniel Macedo Batista¹, and Raian Ali³

¹ Universidade de São Paulo, Brazil
{felipepg, batista}@ime.usp.br

² Universidade de Brasília, Brazil
genaina@cic.unb.br

³ Bournemouth University, UK
rali@bournemouth.ac.uk

⁴ Instituto de Ensino Superior de Brasília - IESB, Brazil

Abstract. Goal-models (GM) have been used in adaptive systems engineering for their ability to capture the different ways to fulfill the requirements. Contextual GM (CGM) extend these models with the notion of context and context-dependent applicability of goals. In this paper, we observe that the interpretation of a goal achievement is itself context-dependent. Thus, we introduce the notion of Pragmatic Goals which have a dynamic satisfaction criteria. However, the specification of context-dependent goals' applicability as well as their interpretations make it hard for stakeholders to decide whether the model is achievable for all possible context combinations. Thus we also developed and evaluated an algorithm to decide on the Pragmatic CGM's achievability. We performed several experiments to evaluate our algorithm regarding correctness and performance and concluded that it can be used for deciding at runtime the tasks to execute under a given context to achieve a quality constraint as well as for pinpointing context sets in which the model is intrinsically unachievable.

Keywords: Context dependency, Goal-models, Requirements Engineering

1 Introduction

Goal-Models (GM) are well established requirements engineering tools to depict and break-down systems using socio-technical concepts [16]. In other words, it provides the goals for which the system should be designed and the various possible ways to reach those goals.

The variability of goal achievement strategies is the baseline for an actor to adapt by deciding which alternative to adopt as a response to certain triggers or adaptation drivers, e.g. faults, errors, availability of computational resources and newly available services and packages. The dynamic environment in which the system operates, *i.e.* its context, could also be an adaptation driver. The Contextual Goal Model (CGM) [1] extends the traditional goal model [5,6,15] with the notion of context. Context may be an activator of goals, a precondition on the applicability of certain alternatives to reach a goal and a factor to consider when evaluating the quality provided by each of these alternatives.

However, we advocate another effect of context on CGMs and requirements in general. The interpretation of a goal achievement is itself context dependent. This means that, in certain contexts, the mere achievement of the sub-goals in a goal model does not imply that the parent goal has been achieved. As an example, consider an ambulance dispatch. The goal of arriving at the patient's location in timely fashion would be seen as achieved when this takes 15 minutes and he/she suffers from dizziness. However, the same goal would not be achieved if the patient suffered from a heart condition. The pragmatism, i.e. dynamic interpretation, is not about the quality but the boolean decision whether a goal is achieved.

In this paper, we introduce the concept of *pragmatic goals* to grasp and model the idea that a goal's interpretation varies according to context. We define the achievability of pragmatic goals as being the capability of fulfilling a goal as interpreted within the context of operation. We also develop and implement an algorithm to compute the execution plan which is likely to achieve a pragmatic goal in a certain context.

In order to validate our approach, we compared the performance and reliability of the answers generated by volunteers and our algorithm [9]. Results showed that volunteers took up to 17 minutes to provide answers with 26.81% reliability without considering our algorithmic approach. This brought to our attention the need for an algorithmic approach. We then evaluated the applicability of our modeling and reasoning algorithm by applying it on a case study of a Mobile Personal Emergency Response System. Finally, we performed a scalability analysis to show the usability of our algorithm in pinpointing context sets in which the CGM as a whole may become unachievable, as well as the possibility of using it to support runtime adaptation by laying out an execution plan which is likely to achieve the necessary constraints. Results show that our algorithm is able to lay out an execution plan in less than a few milliseconds in average. Even in the worst case scenario, for a model with 10000 CGM nodes and 20 contexts, it was able to find a suitable plan in less than 1.2 seconds.

The paper is organized as follows. Section 2 presents the CGM concept on which our model is based. Section 3 presents the pragmatic goals and pragmatic goal achievability concepts. Section 4 presents the proposed model and automated reasoning to decide the pragmatic achievability. Section 5 demonstrates the applicability of the modeling and analysis approach. Section 6 presents related work and Section 7 concludes the paper and outlines our future work.

2 The Contextual Goal-Model

Contextual Goal Model, proposed in [1], explicitly captures the relation between the goal model and dynamic environment of a system. It considers context as an adaptation driver when deciding the goals to activate and the alternatives - subgoal, task or delegation - to adopt and reach the activated goals. Context can also have an effect on the quality of those alternatives and this is captured through the notion of contextual contribution to softgoals.

Context is defined as the reification of the system's environment, *i.e.*, the surrounding in which it operates [8]. For goal models, context is defined as a partial state of the world relevant to an actor's goals [1]. An actor is an entity that has goals and can de-

cide autonomously how to achieve them. A context may be the time of a day, a weather condition, patient’s chronic cardiac problem, etc.

The CGM presented in Fig. 1 depicts the goals to be achieved by a Mobile Personal Emergency Response System which is meant to respond to emergencies in an assisted living environment. The root goal is “respond to emergency”, which is performed by the actor Mobile Personal Emergency Response. The root goal is divided into 4 subgoals: “emergency is detected”, “[p] is notified about emergency”, “central receives [p] info” and “medical care reaches [p]” ([p] stands for “patient”). Such goals are then further decomposed, within the boundary of an actor, to finally reach executable tasks or delegations to other actors. A task is a process performed by the actor and a delegation is the act of passing a goal on to another actor that can perform it.

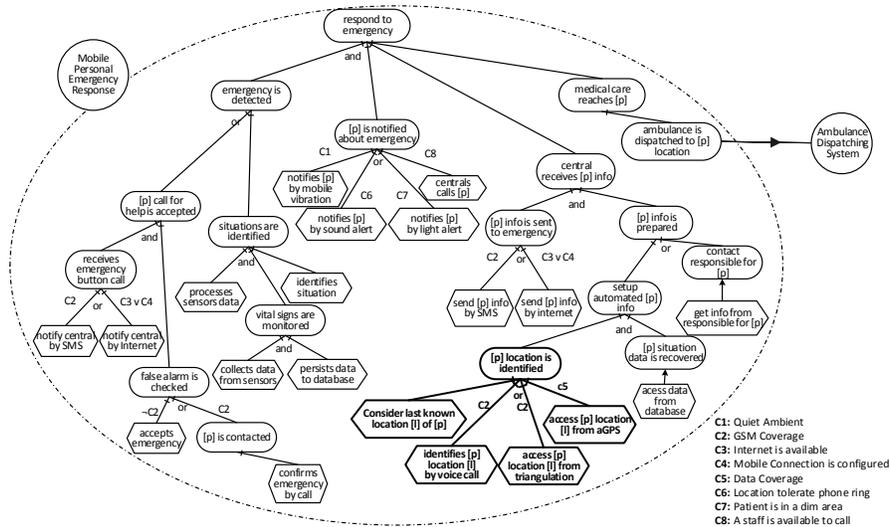


Fig. 1: A CGM for responding to emergencies in an assisted living environment (adapted from [10])

It is important to highlight that not all the subgoals, delegations and/or tasks are always applicable. Some of them depend on certain contexts whether they hold.

3 Conceptualizing Pragmatism in Requirements

Traditionally in a CGM, achieving one (OR-Decomposition) or all (AND- Decomposition) of the subgoals is seen as a satisfactory precondition for achieving the parent goal. We argue that the achievement of some goals would need to be seen from a more pragmatic point-of-view and not as a straightforward implication of the achievement of other goals or the execution of certain tasks. The decision whether a goal is achieved could be context-dependent. Thus we need a more flexible definition of goals to accommodate their contextual interpretation and achievement measures.

The representation of the quality of achievement of a goal as a softgoal is different from the pragmatism of the goal achievement. The pragmatic nature of a goal is not a matter of achieving it with higher or lower quality, but achieving it at all. Also, it has to do with the context at the time of execution and not with the model itself, making the quality requirements more strict or even relaxing them when some contexts apply.

Take the example of Fig. 1: in general, the ambulance may take up to ten minutes to arrive. For a patient with a minor discomfort it can take its time and arrive about ten minutes later without suffering any penalty. On the other hand, if the patient is having a heart attack, one cannot say the goal was achieved. In these situations, the delivered level of quality may not be a separate part from the boolean answer of whether a goal is achieved or not.

A pragmatic goal describes the means to achieve it and also the interpretation of that achievement. This interpretation, which depicts the goal's pragmatic fulfillment criteria, can be expressed as a set of quality constraints (QCs). Unlike softgoals, which are a special type of goal with no clearcut satisfaction criteria[12], these QCs are mandatory and crisp, therefore quantifiable, constraints needed for the fulfillment of a goal and an inherent part of its definition. For instance, take goal "[p] location is identified" from Fig. 1 (G_{loc} for short): it could be defined as "in order to reach G_{loc} , the location must be identified within an error radius of maximum 500 meters and in less than 2 minutes". Again, this would not suffice, as a radius of 500m and 2 minutes might be an over-relaxed condition for patients under critical conditions.

This brings into light another aspect to be taken into account for the pragmatic requirements: the fact that the interpretation for the achievement of a goal is itself context-dependent. We consider that there is a default condition for achieving a goal. However, for specific contexts, we could relax or further strengthen the condition which interprets whether a goal is achieved. We propose that the contextual QCs on the achievement of a goal should be captured together with the other effects of context in the CGM. One advantage of capturing the pragmatic goals within the CGM is to enable reasoning on the possibility of achieving a goal under the current context and QCs. We differentiate these interpretations in the sense that a relaxation condition is not mandatory but a requirement that further strengthen the QC must necessarily be considered.

In the previous example, a QC of getting a location within 500m in less than 2 minutes is a default constraint. However, if the user has access to mobile data (context C5) then a much preciser location can be obtained from the GPS. Under these circumstances, a lock within 500m may seem like an over-relaxed constraint. For a patient with cardiac arrhythmia (context C10), a more strict QC is needed. Suppose that the system has to ensure that an ambulance reaches the patient's home within 5 minutes. Possibly, in this case, a faster but less precise location would be better suited. The requirements for a minor discomfort (context C9) are also more flexible than those for an arrhythmia (C10). In the three specific contexts, the interpretation must differ from the original baseline.

Figure 2 sums up G_{loc} 's interpretation criteria and presents it as a box connected to the goal itself. However, not only the goal's interpretation may vary according to the context but also the tasks' delivered quality of service (QoS) levels. A task may provide different QoS levels when executed in different contexts. This is also represented in Figure 2 by the boxes linked to the tasks.

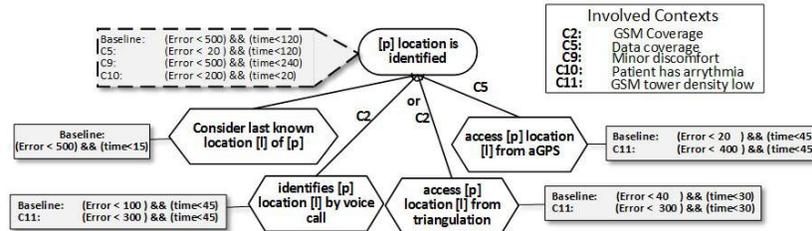


Fig. 2: G_{loc} graphical representation as a pragmatic goal

The set of QCs that represent the pragmatic aspects of a goal and the variable delivered QoS levels extend the traditional CGM into a Pragmatic CGM and allow for the reasoning over the achievability of a goal under a certain context.

Achievability of Pragmatic Goals: Pragmatic goals can only be achieved if their provided QoS levels comply with the QCs specified for them, both of which are context-dependent. This means that we extended the basic effect of context on a CGM to cover success and achievement criteria. Such expressiveness enables further analysis for a key adaptation decision: how to reach our goals while respecting the QCs under the current context where the goals' interpretation, the space of applicable alternatives to reach them and the QoS levels provided by the tasks are all context-dependent?

This part also considers situations where it may not be possible to meet the goal's interpretation QoS standards through any of the applicable sub goals, tasks and/or delegations as they deliver not a static but a context-dependent QoS level. In such cases, we classify the goal as unachievable and the reasoning part can explain the reason.

In the example of Figure 1, if we consider the goal G_{loc} and the contexts impacting on its interpretation, the conclusion is that under a certain context the system may not be able to determine the patient's location with the required precision. This, in practice, does not mean doing nothing. The motivation to do this analysis is because having such knowledge beforehand would allow consideration of other strategies, like adding more alternatives to the same goal to cover a larger range of contexts. At runtime, this conclusion would lead to search for a better variant at a higher-level goal by choosing another branch of an OR-decomposition, which is able to deliver the required quality standard. Therefore, our analysis is both meant for design-time - reasoning to evaluate and validate the comprehensiveness of the solution - and for runtime - searching for the right alternative to reach goals in a specific context.

4 Pragmatic Goal Model

In this section, we concretize our extension to the CGM and elaborate on the new constructs we add as well as their semantics. We mainly enhance the CGM with context-dependent goal interpretations and the expected delivered QoS, which are also context-dependent, for tasks in order to reason about the achievability of the goals for which these tasks are executed.

Figure 3 presents a conceptual model of our extension to the CGM. For the focus of this paper, the CGM could be seen as an aggregation of Requirements. A

Requirement may be specialized into several types: Tasks, Delegations and Goals. A Delegation represents when the Goal or Task (*dependum*) is pursued not by the current but by an external actor (*delegatee*). Tasks are performed by the actor in order to achieve a goal. Tasks may report the expected delivered quality for each metric through the `providedQuality` method. Goals have a refinements set which define the Requirements (subgoals, tasks and/or delegations) that can be used for achieving it as well as a method to distinguish AND- from OR-compositions.

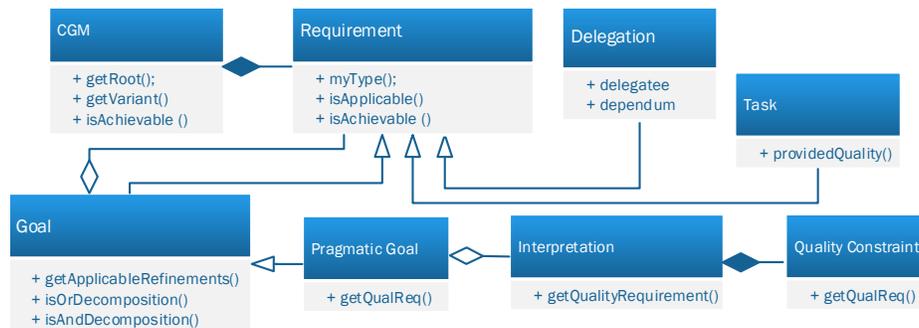


Fig. 3: Conceptual metamodel for our extension to CGMs

Pragmatic Goals extend the Goal concept with a set of Interpretations. A goal Interpretation is an abstract concept that has the function of cross-referencing a context and the appropriate Quality Requirement for that given context. The pragmatic goal is said to be achieved if, and only if, such requirements are met. Otherwise, the goal's delivered QoS is considered inappropriate and the goal is not achieved regardless of achieving one or all of its Requirements.

The Quality Constraints are expressed in terms of the applicableContext in which it holds, the metric that should be considered, the threshold which is a numerical value that represents the scalar value for such metric and the comparison which defines whether the threshold described is the maximum allowed value or the minimum. For instance, to state a quality requirement of at most 250ms for the execution time when context *C1* holds, the metric would be "ms", threshold would be 250, condition would be "Less or Equal" and applicableContext would be *C1*.

Every Requirement inherits the `isAchievable` method. This method can be used either by the final users or by the higher level goals to define whether a particular goal can be achieved for a given quality requirement under the current context. Intermediate goals also have their own predefined interpretation. While this is obviously necessary for the root goal, as the ultimate objective, we also allow certain subgoals to be defined as pragmatic. In principle, actors should be able to impose further constraints on the criteria for achieving any goal within their boundary. The importance of the subgoals quality requirement becomes obvious when dealing with delegation of goals where the external actor may have itself a different, more relaxed or more strict, quality constraint, not necessarily compatible with what the delegator intends.

In this model, both the expectation of delivered quality by the tasks and the quality constraints for the goals, subgoals or delegations are added to the traditional CGM. This is meant to be done by the requirements expert or the domain experts due to the need for specialized knowledge to define such metrics.

Achievability evaluation method To enhance goals with context-dependent interpretation, we must revisit the classical concept of achievability of a goal to fit the nature of Pragmatic CGMs. On top of the basic context effect on a CGM, we enable a higher model expressiveness. Such expressiveness will enable richer adaptation decisions which not only consider the static achievability but also the achievability under the dynamic context and its effect on the fulfillment criteria of a goal. The achievability of a goal and the space of adoptable alternatives to achieve it are essential information to plan adaptation, seen as a selection and enactment of a suitable alternative to reach a goal under a certain context.

To evaluate the achievability of a particular pragmatic goal we present the algorithm in Figure 1. It implements the `Requirement` entity's `isAchievable` method (Figure 3) and correlates three context-dependent aspects from the model: (1) the applicable requirements; (2) the goals' interpretations and; (3) the delivered QoS level provided by the tasks.

The algorithm decides whether the root goal is achievable and, if so, lays out an execution plan, i.e., the set of all tasks to be executed, likely to achieve the desired QCs. The algorithm is recursive with a proven linear complexity with respect to the number of refinements in the CGM [9], building on the fact that the CGM is a tree-structured model without loops and that each refinement may be seen as a tree node.

The algorithm considers the root node of the CGM (line 1) and checks whether the root goal is itself applicable under the current context (line 2), returning `NULL` if it is not (line 3). In the particular case when the variant's root node is a task (line 5) it can readily decide on the achievability. This is because the task nodes know the expected QoS it can deliver for each metric under the context considered in the CGM. By comparing the delivered QoS and required QCs (line 6), the node can decide whether it is capable or not of delivering such QCs. If it can, it will return a plan consisting only of this task (line 7), otherwise it will return `NULL` (line 9) and indicate its inability to fulfill the goal's interpretation.

If the root is not a `Task`, the algorithm will define its quality requirement as the stricter `Quality Constraints` between its own and the QCs passed on as parameters (line 12) and begin laying out an execution plan to fulfill such QCs (line 13). For each of the applicable refinements, it will evaluate if it is achievable (line 16). If the refinement is achievable then, for OR-decompositions, the algorithm returns this plan immediately (lines 18 - 19) and for AND-decompositions it is added to the `complete` plan (lines 21-22). Otherwise, if the refinement is unachievable it will immediately return `NULL` for AND-decompositions (line 25). Finally, for AND-decompositions, should all refinements are achievable it will return the `complete` plan (line 28).

As an outcome, an execution plan is returned for achievable goals. For unachievable goals the `NULL` value is returned to indicate the inability of fulfilling the required constraints, allowing for alternate means of achieving higher level goals to be explored.

Algorithm 1 isAchievable(CGM cgm, Context current, QualityConstraint qualReq)

Require: CGM, current context and desired QCs

```
1: Goal root ← cgm.getRoot()
2: if !root.isApplicable(current) then
3:   return NULL
4: end if
5: if (root.myType() == task) then
6:   if (root.canFulfill(qualReq)) then
7:     return new Plan(root)
8:   else
9:     return NULL
10:  end if
11: end if
12: consideredQualReq ← stricterQC(root.qualReq, qualReq)
13: Plan complete ← NULL
14: deps ← root.getRefinements(cgm, curContext)
15: for all Refinement d in deps do
16:   Plan p ← d.isAchievable(cgm, context, consideredQualReq)
17:   if (p != NULL) then
18:     if (root.isOrDecomposition()) then
19:       return p
20:     end if
21:     if (root.isAndDecomposition()) then
22:       complete ← addPlanToPlan(p, complete)
23:     end if
24:   else if (root.isAndDecomposition()) then
25:     return NULL
26:   end if
27: end for
28: return complete
```

5 Pragmatic Model and Achievability Algorithm Evaluation

To evaluate the need for an algorithmic approach to handle Pragmatic Goals we have conducted a preliminary experiment with volunteers. They were provided with the CGM from Fig. 1 and were asked to check, for a given context set, whether it was achievable or not. The results showed that the volunteers' response had only 26.81% reliability. For the sake of space, we do not report the results of that experiment in this paper, but they can be accessed on [9]. In this Section, we focused on the evaluation of the proposed model's capability to scale over the Pragmatic CGM size with regard to the amount of goals and contexts and its feasibility as a tool to identify unachievable scenarios.

To do so we used the Goal-Question-Metric (GQM) evaluation methodology [4]. GQM is a goal-oriented approach used throughout software engineering to evaluate products and software processes. It assumes that any data gathering must be based on an explicitly documented logical foundation which may be either a goal or an objective.

GQM’s first step is to define high-level evaluation goals. For each goal, a plan consisting of a series of quantifiable questions is devised to specify the necessary measures for duly assessing the evaluation [4]. These questions identify the necessary information to achieve the goals while the metrics define the operational data to be collected to answer each question.

In such a methodology, the main goals of our evaluation are to evaluate: (I) the capability of using our approach for adaptive systems at runtime and; (II) whether it may be used to pinpoint scenarios which render the Pragmatic CGM’s root goal unachievable by construction. From these goals, a GQM plan was defined and is presented in Table 1.

Goal 1: Algorithm’s runtime usage capability	
Question	Metric
1.1 How long would it take to come up with an answer?	Execution time
1.2 How reliable are the answers provided?	% of correct answers
1.3 How does it scale over the amount of goals in the model in average?	Execution time
1.4 How does it scale over the amount of contexts in the model in average?	Execution time
1.5 How does the algorithm scale over the amount of goals in the model in the worst case scenario?	Execution time
1.6 How does the algorithm scale over the amount of contexts in the model in the worst case scenario?	Execution time

Goal 2: Pinpoint unachievable context sets	
Question	Metric
2.1 Can it cover all context sets for models increasingly large models?	Context sets coverage

Table 1: GQM devised plan

Experiment Setup The experiment setup consisted in evaluating the Pragmatic model and the algorithm’s capability to support runtime usage and pinpointing unachievable scenarios. These parts and their evaluations were engineered to provide the metrics demanded by the GQM plan (Table 1).

For the first goal, we evaluated the time to produce an answer, its reliability and a scalability analysis on randomly generated CGMs of different sizes, in terms of goals and contexts amounts. For these time measurements, we have used Java’s `nanoTime()` feature, which has the greatest precision, and considered the average of 100 execution repetitions for each model and context set.

As for the second goal, due to the state explosion problem of possible context sets, we have limited the time amount for each measurement to ten seconds. This way the experiment was faster, and the total amount of time can be easily calculated.

The algorithm from Figure 1 was implemented⁵ using Java OpenJDK 1.7.0_65 and all evaluation tests were performed on a Dell Inspiron 15r SE notebook equipped with an Intel Core i7 processor, 8GB RAM running Ubuntu 14.10, 64 bits and kernel 3.16.0-29-generic. We also used the EclEmma Eclipse’s plugin for ensuring the tests’ code coverage.

All experiments to evaluate the correctness and performance of the algorithm were implemented as automated tests under Java’s JUnit framework. This guarantees that the evaluation is both effortless and repeatable.

Goal 1: Algorithm’s runtime usage capability

Question 1.1: How long would the algorithm take to come up with an answer? To evaluate the time for the algorithm execution on the CGM of Figure 1, we executed 1000 iterations of the algorithm for each context set. The results showed that the algorithm took, in average, less than 1 ms to be executed in each of the 4 scenarios.

Question 1.2: How reliable would an answer provided by the algorithm be? To validate its correctness, we implemented tests for all possible context set for Figure 1. In each context set we have identified all of the inapplicable tasks - both due to context or quality constraints - and asserted that the outputted execution plan did not contain any of these.

Also, we have implemented over 70 tests for the implementation itself. In total, as reported by the EclEmma plugin⁶, this amounted to 95.2% overall code coverage since EclEmma is unable to cover some code lines due to its implementation. Special consideration was given to the Goal, Pragmatic and Refinement classes as well as to the `isAchievable` method which were extensively tested until achieving 100% code coverage (Figure 4). All the tests succeeded thus providing some evidence of the algorithm correctness.

Element	Missed Instructions	Cov.	Missed Branches	Cov.
Refinement		100%		100%
Interpretation		100%		100%
Goal		100%		100%
Plan		100%		n/a
Pragmatic		100%		n/a
CGM		100%		n/a

Fig. 4: Eclipse’s EclEmma plugin reporting 100% code coverage for main classes⁷

One of the purposes of this algorithm is to enable the layout, at runtime, of an execution plan which is able to achieve the CGM’s root goal. To do so, the algorithm needs to be able to process models with varying complexities, both in terms of context

⁵ Source code, evaluation mechanisms, and complete result sets are available at <https://github.com/felps/PragmaticGoals>. Accessed on 2015/03/24

⁶ <http://www.eclEmma.org/>. Accessed on 2015/03/24

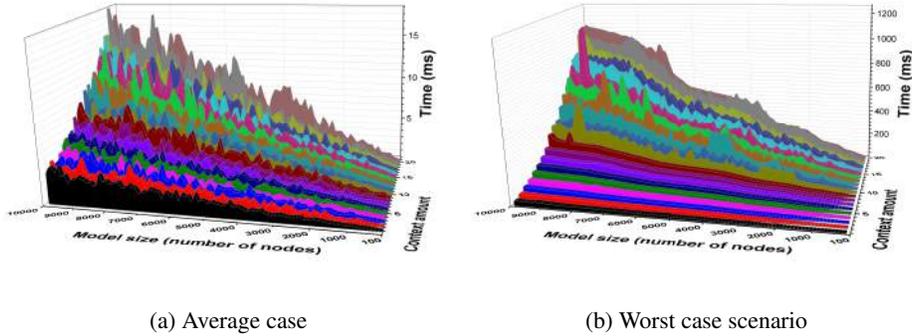


Fig. 5: Algorithm's scalability over the model size, in number of nodes, and context amount

amount and CGM model size, in a reasonable amount of time so that it will not seriously impact the response time.

Question 1.3 and 1.4: How does the algorithm scale over the amount of goals and contexts in the model in average? To evaluate the algorithm's scalability over the model size in terms of goals and contexts amount, we implemented the following test: for each combination of CGM model size (100 to 10000 nodes in steps of 100 nodes) and amount of contexts (1 to 20 contexts), the test would randomly generate 100 CGM models and then the `isAchievable` method was executed 100 times in each model and the average execution time was measured. Finally, it outputs the average execution time for each combination. The resulting average times are presented in Figure 5a.

Question 1.5 and 1.6: How does the algorithm scale over the amount of goals and contexts in the worst case scenario? Similarly to the previous experiment, to evaluate the algorithm's scalability in the worst case we implemented another test. This time the generated model would be the algorithm's worst case scenario: an achievable Pragmatic CGM composed solely of AND-Decompositions. This forces the algorithm to traverse the whole tree. The test generated random Pragmatic CGMs with sizes varying from 100 to 10000 nodes and, for each model, performed 100 executions. The observed average time per execution is shown in Figure 5b.

Analysis of the results With regard to the applicability and reliability, all of our experiments have thoroughly corroborated the proposed algorithm, with execution times of less than 1 ms and no errors in the 10000 repetitions executed on the CGM represented in Fig. 1. In general, as it can be seen from Figure 5, the algorithm's execution time grows linearly over the model's amount of goals and contexts, for the average and for the worst-case scenarios. For models consisting of 10000 nodes and 20 contexts the average time to evaluate the random models was about 18 ms and the worst-case scenario took 1081 ms. This was considered a very good result for such large scenario.

Goal 2: Algorithm’s pinpointing unachievable context sets capability To answer question 3.1 (Can it cover all context sets for models increasingly large models?) we implemented one last test which would generate models varying from 100 to 10000 nodes and a fixed set of 15 contexts. For each model size, 10 random models were generated. Finally, on each of these models and for each combination of the 15 contexts, we have executed the algorithm and measured the percentage of possible context sets it was able to sweep, either finding a suitable solution or not, within the first ten seconds. The results can be seen in Figure 6.

Analysis of the results As shown in Figure 6, on smaller models - up to 300 goals - the algorithm was able to fully analyze the 32768 context sets within the 10 seconds deadline. On larger models - up to 5000 goals - the algorithm was able to analyze around 40% of the combinations. Even at the limit, with models of 10000 nodes, it was able to cover more than 25% of the possible combinations. This result suggests that even for models with up to 10000 goals and 20 contexts the complete analysis can be performed within a minute.

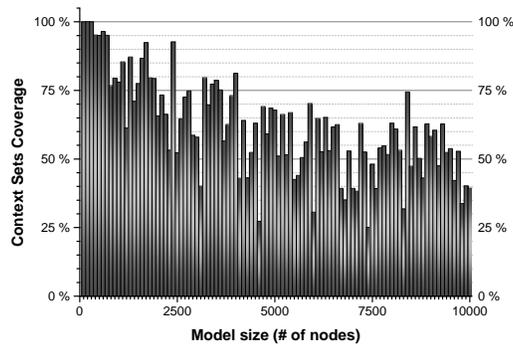


Fig. 6: Percentage of possible context sets analyzed within 10 seconds vs. model size

6 Related Work

Previous work have tackled similar problems but to the best of our knowledge none has dealt with the dynamic context-dependent interpretation of requirements and, in particular, of goals. Relevant approaches include the work of: Sebastiani et al., who map the Goal-Model satisfiability problem into a propositional satisfiability (SAT) problem [11]; Souza and Mylopoulos on Awareness Requirement Goals, that define quality objectives for other goals [12,13]; Baresi and Pasquale on Live Goals: goals whose individual behavior change in order to pursue some qualitative objective and bring the system back to a consistent state [2][3]; Dalpiaz et al. on declarative goals: separate goals whose achievement depends on the effects of its refinements on the environment [7]; and the RELAX framework which provides a more rigorous treatment of requirements explicitly related to self-adaptivity [14].

In essence, related work focus on goal-driven adaptation as a system- or model-wise problem. We argue that the notion of pragmatic goals, introduced in our approach, could enrich the rationale of adaptation proposed in those approaches by treating goal-driven adaptation in a case-to-case context-dependent situation.

The Pragmatic Goals' concept differ from the presented work and from traditional softgoals - which do not have clearcut satisfaction criteria[12] - because, unlike [7], [12] and [13], we consider the pragmatic aspect, *i.e.*, the quality objective as an inseparable part from the goal itself: the mere completion of one or all refinements is not enough to achieve a goal, there may be clients' expectations/demands which must be met and which, differently from [14], is itself context-dependent rather than static. We also deal with the identification and reasoning *a priori* and over the CGM as a whole in an effort to keep the system in a consistent state instead of identifying and correcting inconsistent states like [2] and [3]. Regarding the algorithm itself, our approach enables the algorithm's recursion and achieve linear complexity, by the means of a simplifying assumption: that there are no contributions or denials between different goals. This enabled us to consider the CGM as a tree rather than a generic graph.

Thus, the novelty of our work in comparison to other approaches in requirements-driven adaptation is twofold: (1) The definition of pragmatic goals which means that the satisfaction criteria for goals is context-dependent. (2) The development and implementation of an automated reasoning that can deterministically answer whether the goal is pragmatically achievable and, if it is, point out an execution plan that is likely to achieve it under the current context.

7 Conclusions and Future Work

In this paper we proposed the utilization of a Pragmatic CGM in which the goals' context-dependent interpretation is an integral part of the model. We have also shown why hard goals and softgoals are not enough to grasp some of the real-world peculiarities and context-dependent goal interpretations.

We defined the pragmatic goals' achievability property. A goal's achievability states whether there is any possible execution plan that fulfills the goal's interpretation under a given context. We also proposed, implemented and evaluated an algorithm to decide on the achievability of a goal and lays out an execution plan. We compared the performance of our algorithm to that of a layman's analysis and effectively shown that an algorithmic approach to support the pragmatic goals is needed, considering that human judgment will probably not be fast nor reliable enough. Finally, we performed a scalability analysis on it and shown that it scales linearly over the amount of goals and context amount. We have also shown that, for models up to 10000 nodes and 20 contexts, our algorithm is able to lay out an execution plan in about a second.

For future work, we plan to: (1) integrate this algorithm into a CGM modelling tool; (2) study the possibility of the algorithm to return all task sets instead of a single one and (3) how to enhance the model to integrate task dependencies so that it may represent a context-dependent runtime GM with QoS constraints.

Acknowledgements

This work has been partially funded by CNPq and the EU FP7 Marie Curie Programme through the SOCIAD project.

References

1. Ali, R., Dalpiaz, F., Giorgini, P.: A goal-based framework for contextual requirements modeling and analysis. *Requirements Engineering* 15(4), 439–458 (2010)
2. Baresi, L., Pasquale, L.: Adaptive Goals for Self-Adaptive Service Compositions. 2010 IEEE International Conference on Web Services pp. 353–360 (Jul 2010)
3. Baresi, L., Pasquale, L.: Live goals for adaptive service compositions. In: Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems - SEAMS '10. pp. 114–123 (2010)
4. Basili, V.R., Caldiera, G., Rombach, H.D.: The goal question metric approach. In: *Encyclopedia of Software Engineering*. Wiley (1994)
5. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems* 8(3), 203–236 (2004)
6. Castro, J., Kolp, M., Mylopoulos, J.: Towards requirements-driven information systems engineering: the *Tropos* project. *Information systems* 27(6), 365–389 (2002)
7. Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Adaptive socio-technical systems: a requirements-based approach. *Requirements Engineering* 18(1), 1–24 (Sep 2011)
8. Finkelstein, A., Savigni, A.: A framework for requirements engineering for context-aware services. STRAW 01 (2001)
9. Guimarães, F.P., Rodrigues, G.N., Ali, R., Batista, D.M.: Pragmatic Requirements for Adaptive Systems: a Goal-Driven Modelling and Analysis Approach. ArXiv e-prints (Mar 2015), <http://arxiv.org/abs/1503.07132>
10. Mendonça, D.F., Ali, R., Rodrigues, G.N.: Modelling and analysing contextual failures for dependability requirements. In: Proceedings of the 9th SEAMS. pp. 55–64. ACM, New York, NY, USA (2014)
11. Sebastiani, R., Giorgini, P., Mylopoulos, J.: Simple and minimum-cost satisfiability for goal models. In: *Advanced Information Systems Engineering*. pp. 20–35. Springer (2004)
12. Silva Souza, V.E., Lapouchnian, A., Robinson, W.N., Mylopoulos, J.: Awareness requirements for adaptive systems. In: Proceeding of the 6th SEAMS. p. 60. ACM Press, New York, New York, USA (May 2011)
13. Souza, V.E.S., Mylopoulos, J.: From awareness requirements to adaptive systems: A control-theoretic approach. 2011 2nd International Workshop on Requirements@Run.Time pp. 9–15 (Aug 2011)
14. Whittle, J., Sawyer, P., Bencomo, N., Cheng, B.H., Bruel, J.M.: Relax: Incorporating uncertainty into the specification of self-adaptive systems. In: 17th IEEE RE. pp. 79–88. IEEE (2009)
15. Yu, E.: Modelling strategic relationships for process reengineering. *Social Modeling for Requirements Engineering* 11, 2011 (2011)
16. Yu, E., Mylopoulos, J.: Why goal-oriented requirements engineering. In: Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality. pp. 15–22 (1998)