Evaluating non-functional properties globally

Javier Fernández Briones Alejandro Alonso
Miguel Ángel de Miguel Juan Pedro Silva Gallino
ETS Ingenieros de Telecomunicación
Universidad Politécnica de Madrid
jfbriones@dit.upm.es

Abstract

Real-time systems are usually dependable systems which, besides timing constraints, have to meet some other quality criteria in order to provide certain reliance on its operation. For this reason, a key issue in the development of this kind of system is to trade off the different non-functional aspects involved in the system e.g., time, performance, safety, power, or memory. The success of the development process is often determined by how earlier we can make thorough assumptions and estimations, and hence, take careful decisions about non-functional aspects. Our approach to support this decision activity is based on treating non-functional properties and requirements uniformly, and still supporting specific evaluation and analysis.

1 Introduction

Real-time systems are often dependable systems. Dependable systems are those on which reliance can justifiably be placed on the functionality they deliver. Reliance covers several aspects such as reliability, availability, safety and security; all of them can be characterized using non-functional properties. Engineering systems to the highest reasonably practicable standards of dependability is a great challenge. Failing to do it increases the likelihood of hazards and malfunctions caused by those systems. The consequences range from merely inconveniences, e.g. poor quality in a video application, to the loss of human lives. Only enough confidence can be placed on the system being developed when functional and non-functional specifications are completely defined and a rigorous development process is put in practice.

We claim that preliminary quality assessments should be part of this rigorous process. Within these assessments, engineers study non-functional properties during the design of the system to estimate its probable dependability attributes. The goal is to check whether the system will be able to meet the requirements on those attributes. Additionally, this study will support engineers in recommend quality measures to increase reliance on the system (e.g., supplementary requirements), and optimize the system according to some of the non-functional aspects.

One of the key issues to tackle during the assessment is to trade-off non-functional aspects because they are often in contradiction. For example, there might be no alternative to meet all tasks' deadlines that using a faster processor, but this will increase power consumption and cost. That is why non-functional aspects cannot be considered in isolation and need to be studied simultaneously. This statement becomes more relevant when optimization is involved. During optimization, at the time the value of one non-functional property is improved, other non-functional properties' values are being pushed towards its limits.

Our approach aims at performing preliminary quality assessments that take into consideration all non-functional properties simultaneously, so that we support trade-offs solving.

This work is based on a number of key points: *i*) consider non-functional properties in an homogeneous way: typed variables with a given domain; *ii*) formalize any dependency among non-functional properties explicitly: mathematical statements which define relations on previous variables; *iii*) consider several value specification models for non-functional properties: a steady value, or just a condition on a number of non-functional properties.

Section 2 describes the concept of non-functional property and the uses of the concept, while section 3 shows how we represent and formalize them. Section 4 proposes to quantify and formalize requirements. Section 5 formulates the problem of checking requirements satisfaction. Section 6 proposes another specification model for non-functional properties values, and section 7 reformulates the problem using the new model. Last section provides some concluding remarks.

2 Non-functional properties

We use non-functional properties (NFPs) to represent those system, subsystem, or component attributes which are not needed to implement the desired functionality but, on the other hand, they are required to obtain a quality system and to lead a successful development process. They may describe quality or dependability aspects of such functionality (e.g., security, availability or safety), but also aspects not directly related to the implemented functionality (e.g., adherence to standards and cost). Non-functional aspects have been categorized in different works and standards e.g.: ISO/IEC 9126-1 [1], IEEE Std 830-1998 [2], McCall and Matsumoto quality factors [3], Volere requirements taxonomy [4], or Firesmith requirements taxonomy [5]. Examples of non-functional aspects often under consideration are: availability, interoperability, maintainability, performance, recovery, reliability, scalability, security, safety, or usability. It depends on the domain what non-functional aspects are of interest; for instance, reliability and timing properties are of great important in the embedded systems domain.

There is an existing confusion about the concept non-functional property due to the use of the same concept with different meanings. In fact, the confusion is around the value given to the non-functional property.

- Measured. The value of the property once the system is deployed and in operation. It is an empirical observation, and hence it requires the implementation to be completed; however, measured values of reused subsystems/components can be exploited during the development of new systems. Measured values depend on the instant of time when they are measured and on the way they are measured. They are usually accompanied by an accuracy expression.
- Estimated. The value is just an estimation so the value might not be held by the non-functional property. Estimations are typically made during development and involve making assumptions about the environment, the design, and the components used. Historical data and domain engineers' experience are means to ensure right estimations.
- Calculated. The value is derived from other non-functional properties. A calculated value is often a prediction because it depends on estimated values and/or on imperfect assumptions about how it is derived.

We consider non-functional properties to be fine-grained attributes. What properties are to be used depend on the kind of value. For example, to represent the arrival of events into a system: *i*) the properties for a measured value could be a list of time instants either using absolute time or relative time (time elapsed with the previous event); *ii*) after examining historical data, an estimated value could use the probability mean and the standard deviation of a normal distribution; and *iii*) once some calculations have been performed, we can arrive at e.g., the minimum and maximum inter-arrivals periods or the burst size and period.

Regardless of the kind of value to represent, each non-functional property is a vari-

able with a given type and explicit domain. For instance, maximum and minimum period are positive real numbers; burst size is a positive natural number; or the probability distribution can take one of the literals in the set {normal, exponential, poisson, bernouilli}.

Let x_i be the typed variable representing the non-functional property i and $\mathrm{Dom}(x_i)$ the domain of the variable. Let p_i be a specific value taken for the variable x_i. Let's also define the tuples $\boldsymbol{x}$, to represent the collection of non-functional properties in the system, and $\boldsymbol{p}$, to represent an assignment σ of values to all the properties being considered in the problem.

$$\begin{aligned} \boldsymbol{x} &= (x_1, \ldots, x_I) \\ \boldsymbol{p} &= (p_1, \ldots, p_I) = \sigma(\boldsymbol{x}) \\ p_i &= \sigma(x_i) \in \mathrm{Dom}(x_i) \\ \text{e.g.,} \quad t &= 3 \text{ ms.} \end{aligned}$$

3 NFPs representation

We are only concerned about non-functional properties at development time. We would like to express (reused-)measured values, estimated values and calculated values in an homogeneous way. For this reason, we will express non-functional properties using mathematical statements $O(\boldsymbol{x})$. They are equalities with the non-functional property under consideration at the left side. Measured and estimated values use statements that comprise only one non-functional property. The following are two examples for failure rate and time.

$$\begin{aligned} O_1(r) &\equiv r = 0.001 \\ O_2(t) &\equiv t = 3 \text{ ms} \end{aligned}$$

Calculated values use mathematical statements which depend on several non functional properties. For example, the frequency is the inverse of the period, or the failure rate of a system composed by two components in a pipeline architecture is given by the product of the components' failures rates.

$$\begin{aligned} O_1(f, t) &\equiv f = 1/t \\ O_2(r, r_1, r_2) &\equiv r = r_1 r_2 \end{aligned}$$

Calculated values usually involve complex compositional analysis models, which are not the topic of our work. As an example, the end-to-end latency can only be estimated taking into consideration: components arrangement, processing times and buffering mechanisms. The non-functional aspects with more research are task scheduling, performance and reliability. There is a lot of work trying to predict non-functional properties of an assembly, such as a component-based system, from the known properties of their parts e.g., SEI's work in Predictability by Construction and its initiatives Prediction-Enabled Component Technology (PECT) and Predictable Assembly from Certifiable Components (PACC). Crnkovic et al. [6] classify non-functional properties according to how they can be derived from the properties of the components involved. They can be: directly composable, architecture-related, derived, usage-dependent or system-environment context. However, the authors acknowledge that it might be difficult to model these dependencies. Our approach aims at providing room to all these compositional analysis models.

4 Requirements on NFPs

Whenever possible, requirements have to be expressed quantitatively. Some of the benefits of quantifying requirements are: *i*) to challenge, and thus improve, requirements, *ii*) to formalize requirements in better ways e.g., using mathematical expressions, and *iii*) to enable validation and analysis of requirements. To specify requirements quantitatively: first we have to identify attributes of the requirements that are measurable, and then, to define a metric to express the attributes. In case of non-functional requirements, this means to identify the non-functional properties involve in the requirements. Although quantifying requirements can be difficult, several techniques should help. In [7] two of the best techniques for quantification are discussed: Volere's method [4] and Planguage [8]. The author recognizes that none of them solves all the problems of requirements quantification.

Quantification is very important in the kind of evaluation we propose. Typed variables should enable the use of qualified properties, however we acknowledge the need for a mechanism to determine the order relation of enumeration literals when dealing with optimization.

A requirement comprises one or several non-functional properties. As with non-functional properties, we use mathematical statements to formalize requirements. Requirements are all boolean statements, so they will be also called required constraints. $R(\boldsymbol{x})$ defines the condition to be held by the values of the non-functional properties involved. In other words, an assignment of values $\boldsymbol{p}$ will satisfy the requirement when $R(\boldsymbol{x})$ is true. The relation $R = \{\boldsymbol{p}|R(\boldsymbol{p})\}$ consists of all the assignments that satisfy the condition. The following are two examples of requirement: the first one is a restriction on the power consumption, and the second one relates the inter-arrival time to the execution period.

$$R_1(p) \equiv p \leq 3 \text{ mw}$$
$$R_2(t, T) \equiv t \geq T$$

The concept of requirement on non-functional properties is sometimes differentiated from the concept required non-functional property. This last concept is similar to the concept of required functional interface of a component. There is no difference between the two concepts for the kind of assessment proposed in this paper, and both are requirements to be satisfied.

Sometimes, requirements cannot be fulfilled all together. Altering the system architecture might lead to different compositional analysis models that enable to satisfy the requirements. Nevertheless, this might not be possible and one or more requirements have to remain unfulfilled. This is why a way to categorize requirements should be made available to engineers e.g.: firm/guarantee, soft/best-effort, soft/threshold-best-effort. A better approach would be to quantify the satisfaction level.

5 Problem formulation (I)

Only when system engineers have defined required constraints, they can check whether the constraints are satisfied given the domain of the non-functional properties. Engineers can restraint the search space providing the value specification of some non-functional properties. The problem is two-fold:

1. Is it possible to meet existing requirements knowing there is no value specification for the non-functional properties?

$$\exists \boldsymbol{p} \in \text{Dom}(\boldsymbol{x}) : \wedge_k \boldsymbol{R_k}(\boldsymbol{p})$$

2. Is it possible to meet existing requirements given that every non-functional property is specified?,

$$\wedge_k \boldsymbol{R_k}(\boldsymbol{p}) = \text{true}$$

There are several concepts related to the problem raised:

Valid observation. It stands for any assignment $\boldsymbol{p}$ of values to the non-functional properties that meets the required constraint.

Feasible region. It is the collection of every valid observation.

Optimal configuration. Observations can be compared if there exists a function $f(\boldsymbol{x})|_{\boldsymbol{x} \in S}$ to compare them. If we suppose that the higher the function value the better the observation, the optimal configuration $\boldsymbol{p}^*$ fulfills the following equation:

$$\boldsymbol{p}^* \in S \mid \forall \boldsymbol{p} \in S : f(\boldsymbol{p}^*) \geq f(\boldsymbol{p})$$

6 NFPs specification models

A non-functional property can only hold a value at a given time. However, we have seen that the concept non-functional property is used to represent a promised/potential value. In this section, we would like to draw reader's attention into another type of specification model for the values of non-functional properties. We call the model used so far *single-value*

specification of NFP, and the model in this section *multi-value specification* (they are named asymmetric and symmetric model in [9]).

At development time, it is important not only to be able to specify a single value for a non-functional property but also a set of values for that property i.e., the set of values where the developer promises the actual value will fit in. This is useful for two reasons mainly:

- Actual values are often non-steady and fluctuate during operation. In these situations, engineers are only able to define a range or a condition about the actual values
- We need to be able to express derived properties independently of how the independent variables are specified. For example, the inter-arrival time can be specified using either the minimum and maximum value, or the mean and maximum deviation (jitter). The inter-arrival frequency is the inverse of the inter-arrival time, and we need to specify it independently of the way time is formalized (acknowledging time is a set of allowed values).

We represent the set of allowed values using mathematical relations $O = \{\boldsymbol{x}|O(\boldsymbol{x})\}$. $O(\boldsymbol{x})$ is a boolean mathematical statement that defines the condition we ensure will be held by the value of the non-functional property. An assignment of values $\boldsymbol{p}$ will be allowed when the mathematical statement $O(\boldsymbol{x})$ is true, so $\boldsymbol{p}$ belongs to the relation O. We call $O(\boldsymbol{x})$ offered or provided constraint. Offered constraints can be used to express calculated values since they can involve several non-functional properties.

The following is an example of single-value specification of non-functional properties. The main problems of this specification model is: verbosity and usability.

$$O_{1a}(t_{min}) \equiv t_{min} = 2.9\text{ms}$$
$$O_{1b}(t_{max}) \equiv t_{max} = 3.1\text{ms}$$
$$O_{2a}(t_{max}, f_{min}) \equiv f_{min} = 1/t_{max}$$
$$O_{2b}(t_{min}, f_{max}) \equiv f_{max} = 1/t_{min}$$

And the following is an example of multi-value specification of non-functional properties.

$$O_1(t) \equiv 2.9\text{ ms} \leq t \leq 3.1\text{ ms}$$
$$O_2(t, f) \equiv f = 1/t$$

7 Problem formulation (II)

The use of the multi-value specification model demands to change the formalization of the problem. Now, when a property is specified, required constraints have to be satisfied for all the values meeting that value specification. We need to ensure the satisfaction of all the required constraints for every potential value the property might hold. The second formulation of the problem becomes:

$$\forall \boldsymbol{p} \in \text{Dom}(\boldsymbol{x}) : \wedge_j \boldsymbol{O_j}(\boldsymbol{p}) \Rightarrow \wedge_k \boldsymbol{R_k}(\boldsymbol{p})$$

This constraint is also valid for single-value specifications: it degenerates into ($\wedge_k \boldsymbol{R_k}(\boldsymbol{p})$ = true) in case of using single-value specifications only (there will be only one $\boldsymbol{p}$ to check since every variable has only one assigned value)

8 Concluding remarks

We have formulated the problem of assessing non-functional properties globally as opposed to evaluating each non-functional aspect individually. Dealing with (and formalizing) value specifications and requirements on non-functional properties uniformly should be a good step forward to leverage generic tools (instead of current ad-hoc developments). However, we recognize that we do not ease the evaluation of the properties and we only support decision-making about existing trade-offs between the different aspects.

After formulating the problem raised in this paper, it is possible to study different solver alternatives. The type and complexity of the statement $O_j(\boldsymbol{x})$ and $R_k(\boldsymbol{x})$ are key issues to decide the solver to use. The use of a single-value specification model makes the problem more tractable because evaluating for a single value is easier than for a set. As an example,

linear programming techniques could be of use if mathematical statements comprise only linear functions and inequations.

Optimization is only possible when there is a at least one unspecified variable (one degree of freedom). The problem of providing value specifications for non-functional properties is that it might leave no room to optimization. Some of our previous work [10, 11, 12] tackles alternative value specifications i.e., given a number of value specifications from the domain how to choose the one that satisfies the requirements. These choices can be used to represent design alternatives, deployment options, operation modes or entities adaptability. At the design-phase, they all can be considered the same regarding our goal: the assessment of non-functional requirements.

References

[1] "ISO/IEC 9126–1: software engineering — product quality — part 1: Quality model," International Organization for Standardization/International Electrotechnical Commission, Standard 9126-1, June 2001.

[2] "IEEE Std 830: recommended practice for software requirements specifcations," Institute of Electrical and Electronics Engineers, IEEE Standard 830, June 1998.

[3] J. A. McCall and M. T. Matsumoto, "Software quality measurement manual," General Electric Company and Rome Air Development Center, Tech. Rep. RADC-TR-80-109 Part II, April 1980.

[4] S. Robertson and J. Robertson, *Mastering the Requirements Process.* Addison-Wesley Professional, Aug. 1999.

[5] D. G. Firesmith, "Common concepts underlying safety, security, and survivability engineering," Carnegie Mellon Software Engineering Institute, Technical Note CMU/SEI-2003-TN-033, December 2003.

[6] I. Crnkovic, M. Larsson, and O. Preiss, "Concerning predictability in dependable component-based systems: Classification of quality attributes," in *Architecting Dependable Systems III*, vol. 3549. Springer Berlin / Heidelberg, 2005, pp. 257–278.

[7] N. Maiden, "Improve your requirements: Quantify them," *IEEE Software*, vol. 23, no. 6, pp. 68–69, 2006.

[8] T. Gilb, *Competitive Engineering: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage.* Butterworth-Heinemann, Aug. 2005.

[9] A. R. Cortés, O. Martín-Díaz, A. D. Toro, and M. Toro, "Improving the automatic procurement of web services using constraint programming," *Int. J. Cooperative Inf. Syst.*, vol. 14, no. 4, pp. 439–468, 2005.

[10] J. F. Briones, M. A. de Miguel, A. Alonso, and J. P. Silva, "Modeling quality of service adaptability," in *Enterprise Distributed Object Computing Workshops (Advances in Quality of Service Management), International Conference on*, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 50–27.

[11] ——, "Quality of service composition and adaptability of software architectures," in *12th IEEE International Symposium on Object component service-oriented Real-time distributed Computing*, 2009.

[12] J. F. Briones, M. A. de Miguel, J. P. Silva, and A. Alonso, "On the requirements for quality composability modeling and analysis," in *MoBE*, 2010.