Dealing with Non-Functional Requirements in Model-Driven Development: A Survey

David Ameller, Xavier Franch, Cristina Gómez, Silverio Martínez-Fernández, João Araújo, Stefan Biffl, Jordi Cabot, Vittorio Cortellessa, Daniel Méndez Fernández, Ana Moreira, Henry Muccini, Antonio Vallecillo, Manuel Wimmer, Vasco Amaral, Wolfgang Böhm, Hugo Bruneliere, Loli Burgueño, Miguel Goulão, Sabine Teufl, Luca Berardinelli

Abstract—*Context*: Managing Non-Functional Requirements (NFRs) in software projects is challenging, and projects that adopt Model-Driven Development (MDD) are no exception. Although several methods and techniques have been proposed to face this challenge, there is still little evidence on how NFRs are handled in MDD by practitioners. Knowing more about the state of the practice may help researchers to steer their research and practitioners to improve their daily work. *Objective*: In this paper, we present our findings from an interview-based survey conducted with practitioners working in 18 different companies from 6 European countries. From a practitioner's point of view, the paper shows what barriers and benefits the management of NFRs as part of the MDD process can bring to companies, how NFRs are supported by MDD approaches, and which strategies are followed when (some) types of NFRs are not supported by MDD approaches. *Results*: Our study shows that practitioners perceive MDD adoption as a complex process with little to no tool support for NFRs, reporting productivity and maintainability as the types of NFRs expected to be supported when MDD is adopted. But in general, companies adapt MDD to deal with NFRs. When NFRs are not supported, the generated code is sometimes changed manually, thus compromising the maintainability of the software developed. However, the interviewed practitioners claim that the benefits of using MDD outweight the extra effort required by these manual adaptations. *Conclusion*: Overall, the results indicate that it is important for practitioners to handle NFRs in MDD, but further research is necessary in order to lower the barrier for supporting a broad spectrum of NFRs with MDD. Still, much conceptual and tool implementation work seems to be necessary to lower the barrier of integrating the broad spectrum of NFRs in practice.

Index Terms—Model-Driven Development, Non-Functional Requirements, Quality Requirements, Requirements Engineering, Survey.

1 Introduction

Model-Driven Development (MDD) refers to the systematic use of models as first-class entities throughout the software engineering lifecycle [1], [2]. Its objective is to increase productivity, portability and reuse, and to reduce time to market by enabling complex systems development using models. Models are defined with concepts that are much less bound to the underlying implementation technology, thus abstracting from unnecessary details. This favours the transformation of such models into "real things" [2].

The confidence in the practical relevance of MDD is strengthened by ongoing technology projects (*e.g.,* the Eclipse Modelling Project[1]), industrial practices [3], [4], and reported success stories [5]. Obstacles and challenges to be surpassed have also been studied [6].

When analysing the literature related to the industrial adoption of MDD, one aspect becomes evident. While Non-Functional Requirements (NFRs) are becoming more and more important in industry (*e.g.*, in relation to quality assurance, or when checking compliance to security regulations) and MDD is being used in critical domains such as automotive, defence, and aerospace, there is still a lack of evidence on how NFRs are managed and fullfilled by MDD processes. In other areas, such as Service-Based Systems, there is scarce NFRs support [7]. This knowledge gap was already reported in 2010 [8] and the situation has not improved significantly since then.

Under the premise that practitioners need to consider NFRs regardless of the concrete software development approach they follow, we conducted an industrial survey based on interviews to better understand: 1) the importance given to NFRs in comparison to functional requirements and the NFRs expected to be satisfied when MDD is adopted, 2) the NFRs really satisfied when MDD is adopted, the development stages in which they are addressed and the techniques used, and if the companies need to tailor their MDD approaches to guarantee those NFRs, and 3) the strategies used by the companies when NFRs are unsupported by MDD approaches. The interview-based survey follows Ciolkowski *et al.* guidelines [9]. We interviewed practitioners from 18 companies that use MDD in their software projects (5 medium-sized and 13 large companies) in 6 European countries, recruited by researchers of each participating country. The analysis of the answers reveals that: 1) NFRs are less important than or equally important as functional requirements, 2) productivity, maintainability and reusability are the NFRs most expected to be improved before adopting MDD, 3) the expectations are met for productivity and maintainability when MDD is adopted, 4) modelling, transformation functions and MDD adaptations are used for specifying and satisfying NFRs and 5) manual

• *D. Ameller is with the Univ. Politècnica de Catalunya, Barcelona, Spain. E-mail: dameller@essi.upc.edu*

Manuscript received Month DD, YYYY; revised Month DD, YYYY.

1. https://www.eclipse.org/modeling

changes, which may compromise maintenance, or even discarding NFRs, are the strategies followed when NFRs are unsupported by MDD approaches.

The pieces of evidence collected in our survey represent a new asset in the software engineering body of knowledge. The findings we drew from the way practitioners deal with NFRs when they use an MDD approach, the alignment of the survey findings with existing results from literature, and the discussion of possible actions that could help improving the state of the practice in the field could help practitioners and researchers improving their daily work. For instance, practitioners may discover that there exist research results that allow for including NFRs into the MDD processes in a systematic way. Moreover, researchers may become aware of what types of NFRs practitioners perceive as worst supported in their projects and focus their efforts in their types.

The paper is structured as follows. Section 2 provides the background. Section 3 reports related work. Section 4 describes the methodology used. The results are reported in Section 5 and discussed in Section 6. Section 7 discusses the threats to validity before providing concluding remarks and an outline of a research agenda in Section 8.

2 Background

2.1 Model-Driven Development

MDD can be defined as *"a development paradigm that uses models as the primary artefact of the development process"* [10]. Other concepts related to MDD are: 1) *Model-Driven Engineering* (MDE), which is a systematization of MDD to all software engineering activities (*e.g.,* forward engineering, reverse engineering, software evolution, and systems interoperability); 2) *Model-Driven Architecture* (MDA) [11], which is the *Object Management Group* (OMG) approach to MDD. They all imply the use of models as first-class entities and the general consideration is that they provide a way to design and develop software systems. Unless otherwise specified, in this paper we will focus on MDD.

There are two key concepts in MDD:

- *Model*. Two types of models are mostly considered in practice: terminal models (also sometimes called object models) (M1) and metamodels (M2) [12], [13]. However, in general, any number of modeling levels may exist [14]. A terminal model (or a set of terminal models) is a representation of a system/domain capturing some of its characteristics by means of abstraction. A terminal model is expressed using a precise modelling language and conforms syntactically to a metamodel that defines the kind of model elements (and their relationships) that may appear in the model. This way, metamodels play a similar role than grammars in the domain of programming languages.
- *Model transformation*. Model transformations take one or more terminal models as input and generate one or more terminal models (model-to-model transformations) or textual artefacts (model-to-text transformations) as output. The input and output models may conform to the same metamodel or to different ones. Example of such transformations are those defined to produce performance models (*e.g.,* Queueing Networks for performance analysis) from software models, or as part of code-generation processes.

The adoption of MDD has the potential to bring along many benefits such as improvements in the productivity and maintainability of the system [15] or as mentioned by many practitioners, the ability to represent the software system at an abstract level (*i.e.,* technology- or platform-independent). The adoption of MDD involves the use of some modelling language, such as the UML, to represent the system, and the use of one or more model transformations to generate artefacts (*e.g.,* source code, documentation). However, some companies have a more in-depth adoption, which also involves the use of customized metamodels, such as extensions for the UML metamodel [16], or the use of *Domain-Specific Languages* (DSLs) [17] tailored to solve specific kinds of problems.

2.2 Non-Functional Requirements

Although there is still no consensus on the definition of Non-Functional Requirements (NFRs) [18], there is a general agreement that NFRs refer to quality attributes or constraints on the operation, use or development process of a software system. Therefore, NFRs are "requirements which are not specifically concerned with the functionality of a system" [19], frequently stated informally during requirements analysis, [which] are often contradictory, difficult to enforce during software development and to validate [20]. Examples include environmental and implementation constraints which are not specifically concerned with the functionality of a system, performance, platform dependencies, maintainability, extensibility, reliability [20], user friendliness and hedonic qualities [21], hence comprising quality attributes and other system constraints. NFRs often tend to interact or to conflict with functional requirements and other NFRs. Trade-off analysis must then be conducted to resolve the conflicts.

There are many taxonomies or otherwise classifications for NFRs. For example, Roman [22] presents a taxonomy that includes interface, performance, operation, lifecycle, economic, and political constraints. Somerville [23] classifies NFRs as product, organizational, and external requirements. The ISO/IEC 25010 standard [24] is arguably the most widespread classification used. It has to be noted that seven out of the eight main software product quality categories defined by ISO/IEC 25010 are dealing with NFRs, *e.g.,* performance efficiency, compatibility, security to mention just a few. Furthermore, these seven categories provide over 30 subcategories for NFRs.

Software companies are increasing their awareness of the importance of NFRs in the construction and evolution of large and complex software-intensive systems. NFRs are decisive for the success or failure of such systems. The lack of integration of NFRs with functional requirements can result in long time-to-market and cost overruns in software projects [20], [25], [26], [27]. Indeed, some researchers have already reported that the notion of NFR is not always clear for practitioners (*e.g.,* Ameller *et al.* [28]). Sometimes, the problem is of terminological nature (*e.g.,* the meaning of

"dependability" or "performance" could be quite diverse) but it can also be of conceptual nature (*e.g.*, some respondents in this study have qualified "productivity" as an NFR while others have not). Any industrial study involving the concept of NFR needs to be aware of these problems and decide how to mitigate them. In our study, to avoid any exclusion, we consider as (type of) NFRs anything referred to as such by the participants (including aspects related to the development process, *e.g.*, productivity).

2.3 Non-Functional Requirements in MDD

Several research approaches have been proposed to deal with NFRs in MDD. The common approach, in the state of the art, is to add NFRs to models by using UML profiles and, in fewer cases, NFRs are added directly to the UML models (*e.g.*, as a comment) or by providing a whole metamodel designed for this objective [29]. The earliest contributions appeared at the beginning of the 2000s in the domain of performance validation (*e.g.*, the *Workshop on Software and Performance* [30]). They were soon followed by MDD approaches in reliability and safety domains (*e.g.*, Cortellessa *et al.* [31] and Grunske *et al.* [32]). A research effort was conducted thereafter to define frameworks for the systematic application of MDD approaches [33], independently of the notations and the processes adopted. More recently, MDD approaches were introduced to extend the targeted NFRs (*e.g.*, to guarantee security during software evolution [34]), to cope with specific paradigms (such as Service-Oriented Architectures [35]) and/or specific application domains (such as Adaptive Systems [36]) and to control the fulfilment of NFRs in model transformations [37].

3 Related Work

Several empirical studies exist on the state of the practice in MDD (*e.g.*, Hutchinson *et al.* [38]) or NFRs (*e.g.*, Ameller *et al.* [28]) but not exploring the state of the practice on MDD and NFRs together. In very few cases, the studies about MDD mention requirements but their focus is mostly on functional requirements. Similarly, a few studies about NFRs mention the use of models, but none as part of an MDD approach. Therefore, our study will cover an unexplored area, that is, how companies deal with NFRs when using MDD.

Table 1 summarizes the studies found in literature related to NFRs or to MDD. Due to their very different nature, we distinguish between surveys, using either interviews or online questionnaires for data extraction, and case studies. In some cases, the studies used a combination of different types of empirical studies (*e.g.*, Hutchinson *et al.* [38]) or different types of data extraction (*e.g.*, Mohagheghi *et al.* [4]).

The interest of the studies can be quite different. While some are very generic, *e.g.*, for MDD [39] and for NFRs [40], others target some particular aspect. For instance, they may focus on acceptance in industry [4], productivity [41], [42], tool-related issues impacting MDD adoption [43], [44], viewpoints from a particular role like software architect [28], [45], [46], etc. It is worth to remark that some surveys have been conducted in a particular geographical area, *e.g.*, Italy [47] or Brazil [48].

In this section we do not provide the details of the studies. Instead, we discuss the significant connections of these works with the results of our study in Section 6 once the results of our own study have been presented.

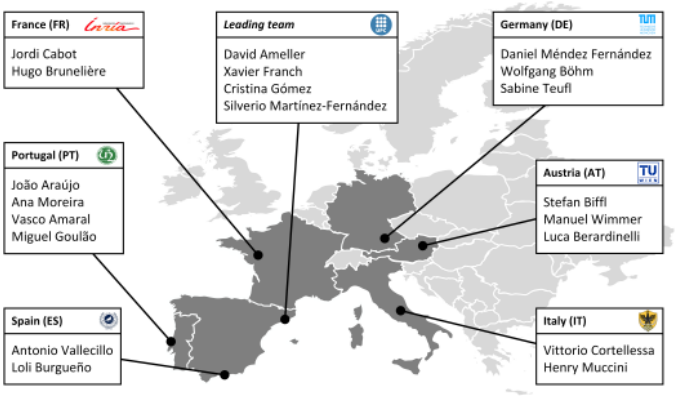


Fig. 1. Participating Countries and Researchers

4 Methodology

We followed Ciolkowski *et al.'s* guidelines [9] for conducting empirical studies based on surveys. These guidelines identify six steps: definition, design, implementation, execution, analysis, and packaging (see Figure 2).

We produced a protocol document detailing these six steps[2]. All the involved researchers, the authors of this paper and the researchers in the acknowledgements, validated this protocol based on their experiences in similar empirical studies. Furthermore, we published a peer-reviewed paper in RE@Next [55] with the contents of this protocol.

We defined three RQs divided into ten sub-RQs (see Table 2). However, we removed RQ1.1 (about the context of use of MDD in companies) because it is not directly related to NFRs. Still, since we collected data for this sub-RQ, we can refer to this data when contextualizing some other finding in the discussion. Therefore, RQ1 in the paper includes RQ1.2 and RQ1.3 from the protocol. We also removed RQ2.2 (about NFR characteristics) and RQ2.3 (about tools) because we did not collect enough data to provide insightful results. Therefore, RQ2 in the paper only includes RQ2.1 and RQ2.4 from the protocol. RQ3 keeps the three sub-RQs from the protocol (RQ3.1-3.3). The resulting three RQs are shown in Table 3 and discussed in detail in Section 4.2.

In the following subsections, we provide a summary of the most relevant aspects of the methodology. For further details, we refer to the protocol document (see Footnote 2).

4.1 Goal of this Study

Following the GQM (Goal-Question-Metric) technique [56], we defined the goal of our study as follows:

- *Purpose.* Explore, analyse, and characterize
- *Object.* MDD approaches
- *Focus.* Deal with NFRs
- *Viewpoint.* Technology experts in the company
- *Context.* Companies using MDD in real projects

The expected outcome of this survey is a better understanding on how companies handle NFRs when they

2. Protocol: https://figshare.com/s/cceb874c87b5bd5213ce

TABLE 1
Characteristics of the Related Empirical Studies

Ref.	*Type of study*	*Data extraction***	*Area*	*Topic*	*Sample****	*Year of publication*
[43]	Survey and case study	Int.	MDD	Tool-related issues	19 pract. / 2 org.	2017
[44]	Survey	Int.	MDD	Software evolution	3 org.	2016
[49]	Survey	Quest.	MDD	Embedded systems	113 pract.	2016
[50]	Survey	Quest.	NFR	Service-based systems	56 pract.	2016
[45]	Survey	Int.	NFR	Contract-based projects	21 pract.	2015
[46]	Survey	Int. and quest.*	NFR	Specification and validation	6 org.	2014
[38]	Survey and case study	Quest. and int.	MDD	Factors for success and failure	449 pract. / 4 org.	2014
[4]	Survey	Int. and quest.	MDD	State of the practice	4 org.	2013
[47]	Survey	Quest.	MDD	Benefits and drawbacks	155 pract.	2013
[48]	Survey	Quest.	MDD	Embedded systems	209 pract.	2013
[51]	Case study	Int.	MDD	Benefits	3 org.	2013
[28]	Survey	Int.	NFR	State of the practice	12 org.	2012
[39]	Survey	Int. and quest.	MDD	State of the practice	22 pract. / 250 pract.	2011
[40]	Survey	Int.	NFR	Embedded systems	10 pract.	2009
[41]	Case study	Not specified	MDD	eBusiness software	1 org.	2007
[42]	Case study	Int. and quest.*	MDD	MDD industrial infusion	1 org.	2006
[52]	Case study	Not specified	MDD	Large IT consultancy context	1 org.	2006
[53]	Case study	Not specified	MDD	Large industrial context	1 org.	2005
[54]	Survey	Int.	NFR	NFR-related problems	2 org.	2003

* These works aggregate the results from different data extraction mechanisms
** Int.: Interviews; Quest.: Questionnaires
*** Pract.: Practitioners; Org.: Organizations

TABLE 2
Original RQs as Defined in the Protocol

Id	*Research Question*
RQ1	**In which context is MDD adopted by companies?**
RQ1.1	What factors motivate or discourage the adoption of MDD?
RQ1.2	Which types of NFRs are linked to these factors?
RQ1.3	To what extent are NFRs relevant for those projects that adopt MDD?
RQ2	**To what extent do MDD approaches adopted by companies support NFRs?**
RQ2.1	Which types of NFRs are supported by the adopted MDD approaches?
RQ2.2	Which characteristics do these NFRs exhibit?
RQ2.3	Which notations and tools are used for the supported types of NFRs?
RQ2.4	At which stages of the adopted MDD approach are these NFRs handled?
RQ3	**How do companies deal with NFRs when the adopted MDD approach does not support them?**
RQ3.1	How are MDD approaches customized to take into account the previously unsupported types of NFRs?
RQ3.2	How do companies deal with an NFR which is not supported by MDD?
RQ3.3	To what extent are the drawbacks of dealing with unsupported types of NFRs compensated by the benefits of adopting MDD?

use MDD for software production. Furthermore, from the results obtained, we have identified and discussed several findings and gave our opinion about them (*i.e.,* how the scientific community and practitioners can help improve the current state of the practice).

TABLE 3
Modified RQs

Id	*Research Question*
RQ1	**What benefits can the management of NFRs bring to companies as part of their MDD processes?**
RQ1.1	Which types of NFRs are linked to the factors that motivate the adoption of MDD? (former RQ1.2)
RQ1.2	To what extent are NFRs relevant for those projects that adopt MDD? (former RQ1.3)
RQ2	**How do MDD approaches adopted by companies support NFRs?**
RQ2.1	Which types of NFRs are supported by the adopted MDD approaches?
RQ2.2	At which stages of the adopted MDD approach are these NFRs handled? (former RQ2.4)
RQ3	**How do companies deal with NFRs when the adopted MDD approach does not support them?**
RQ3.1	How are MDD approaches customized to take into account the previously unsupported types of NFRs?
RQ3.2	How do companies deal with an NFR which is not supported by MDD?
RQ3.3	To what extent are the drawbacks of dealing with unsupported types of NFRs compensated by the benefits of adopting MDD?

4.2 Research Questions

Based on the goal of the study, we report on the answers to the following three research questions.

RQ1 (*what benefits can the management of NFRs bring to companies as part of their MDD processes?*) aims at understanding the importance of NFRs in companies that adopt MDD. Firstly, we want to know the level of importance given to NFRs in comparison to functional requirements. Secondly, we aim to discover the NFRs expected to achieve

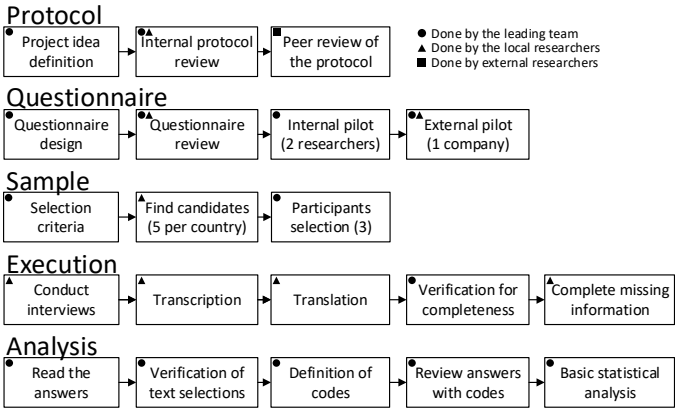


Fig. 2. Description of the Protocol Defined Process

when MDD is adopted and which of them influence the adoption of an MDD approach. This RQ is answered in the basis of several questions in the questionnaire, for instance "How relevant are NFRs in the projects that well-fit with MDD?" and "Which were the relevant NFRs of the [selected] project?".

RQ2 (*how do MDD approaches adopted by companies support NFRs?*) focuses on understanding the current level of support of NFRs in the adopted MDD approaches. Firstly, we want to know if companies need to tailor their MDD approaches to support specific types of NFRs, or if there are some types of NFRs that are directly supported, *i.e.*, without modifying their MDD approaches. Secondly, we want to determine in which part of the MDD process the supported NFRs are addressed and what techniques are used to this end. In knowing how NFRs are supported by MDD, we base our observations in questions like "Which NFRs are supported [by MDD as it is now in the company]" and "Please, provide an example of one particular NFR supported by the MDD".

RQ3 (*how do companies deal with NFRs when the adopted MDD approach does not support their management?*) aims at understanding the strategies used by the companies when they have to cope with those types of NFRs that are unsupported by the MDD approaches adopted in their contexts. There are two envisioned strategies [8]: to adapt the MDD process so that it can account for the unsupported types of NFRs, or to manually adjust the resulting artefacts of the MDD process to satisfy the unsupported NFRs. However, we expect that companies may very well use other strategies.

4.3 Research Team

The study was initiated by the research group at the Universitat Politècnica de Catalunya (UPC) who issued invitations to the rest of the groups of the other institutions. We wanted to involve a large number of teams for two reasons: 1) to form a robust research team composed of expert researchers from the two main areas of research (MDD and NFRs) plus the added value of their experience with empirical studies; 2) to facilitate reaching a diverse sample of industrial participants. UPC selected the invited teams from their contact network. In a few cases, previous collaborations existed, but for most teams this was the first joint endeavour. Initially, eight teams joined the study, but two of them withdrew in the course of the study, yielding to the final six participant teams (see Figure 1).

Along the paper, when necessary, we distinguish among the leading team (the first four authors of the article, who are from the UPC team) and local researchers (the researchers from each of the six participating countries). For example, local researchers executed the interviews while the leading team performed the data analysis.

4.4 Population and Survey Sample

The main entities of this study are the technology experts from the participating companies. We did not restrict the participating companies concerning their size or application domain, but we required them to have experience using MDD in their software projects. Therefore, our target participant, is a highly skilled MDD practitioner.

For the sample selection, each team of local researchers provided a list of 5 tentative companies, from which the leading team selected 3 companies based on their adequacy for the study (*i.e.*, direct participation in MDD projects that included models, metamodels, and transformations). The invitation letter used to contact potential participants is available online[3]. In total, 18 companies participate (see Table 4) which seems a reasonable number especially compared to the other related studies (see Table 1). The companies have different levels of MDD adoption. For each company, we interviewed one employee. Most of the interviewees had a software engineering background (14), while dominant academic degrees were MSc (10) and PhD (6).

4.5 Questionnaire Design

We designed a questionnaire[4] based on Dillman *et al.* 's [57] recommendations to gather the data required to answer the RQs. It consists of 10 questions and 43 sub-questions along with the instructions for how to conduct the interview. Among them, 32 are open questions, while the remaining ones have a set of predefined answers. The idea was to design a questionnaire with a well-defined structure, but leaving most of the questions open to let the interviewers and interviewees go in-depth when some interesting topic arose during the interview (*i.e.*, a semi-structured interview). Among the 10 main questions, three were used to understand the context of the companies (questions 1–3) and one (question 6) to validate the previous responses given and to put the participants in context for the following questions.

The initial set of questions was proposed by the leading team, then reviewed by the local researchers, and last validated in two pilot interviews.

Since the local researchers were responsible for the survey execution, the leading team included precise and detailed guidelines as part of the questionnaire with instructions for the interviewers on how to ask each question. For example, for the first question, the first instruction was to let the participants freely explain what MDD was for them and then try to match the provided definition with one of

3. Invitation letter: https://figshare.com/s/cceb874c87b5bd5213ce
4. Questionnarie: https://figshare.com/s/cceb874c87b5bd5213ce

TABLE 4
Survey Sample

Id	*Domain*	*Size*[1]	*Years*[2]	*Projects*[3]	*Level*[4]
FR1	Digital systems	Large	17	>10	L4
FR2	Electronics	Large	3	1	L3-L4
FR3	IT services	Large	15	15	L3-L4
PT1	Aerospace and defence	Large	2	1	-
PT2	Mobile and web	Large	2	2	L3-L4
PT3	Electronics	Medium	6	>5	L5
AT1	Software tools	Medium	2	1	L4
AT2	Software tools	Large	>20	>10	L4
AT3	Software tools	Medium	10	30	L2-L3
ES1	IT services	Large	6	10	L4
ES2	IT services	Large	7	4	L4
ES3	IT services	Medium	9	10	L4
DE1	IT services	Large	3	15	L3
DE2	Defence	Large	17	4	L3
DE3	Data processing	Large	7	5	L4
IT1	Aeronautics	Large	10	>20	L1-L2
IT2	Defence	Large	2	2	L1-L2
IT3	Mobile and web	Medium	14	10	L5

[1] Medium: Less than 100 employees; Large: 100 or more
[2] Years of the interviewee using MDD
[3] Projects in which the interviewee used MDD
[4] L1 *Ad hoc modelling*: Modelling practices are rarely used.
L2 *Basic MDD*: Models with business and technical concepts. Code is obtained, manually or automatically, from models.
L3 *Initial MDD*: Business models are manually converted to technical models which are converted to code automatically.
L4 *Integrated MDD*: Use of DSLs. Domain, business and technical concepts separated. Reusable infrastructure technical model. Models used for testing and correcting design issues.
L5 *Ultimate MDD*: Models are executable. Reusable system family engineering mind-set with common set of MDD assets (transformations, domain models, metamodels, ...).

the listed options. Our intention with the guidelines was to reduce the interviewers' bias and to guarantee the capability to aggregate the data gathered for the analysis.

4.6 Survey Execution

Local researchers used face-to-face interviews in most cases except for 6 phone interviews. Three interviews were recorded and transcribed verbatim, the rest was summarised (nine very detailed close to verbatim, six focussing on the essentials) based on recordings. When necessary, the local researchers translated the obtained answers to English. Furthermore, the leading team validated the interviews for completeness and adequacy (*e.g.*, if some answers were incomplete or needed additional details, the local researchers provided the missing information) and the interviews were also reviewed by the interviewees. The transcriptions/summaries of the interviews are available online[5].

The duration of the interviews had an average of 75 minutes with a minimum of 44 and a maximum of 98 minutes. The conducted interviews took place from June to November 2015.

5. Dataset: https://figshare.com/s/cceb874c87b5bd5213ce

4.7 Data Analysis

Since the majority of questions were open, we decided to apply qualitative content analysis [58] to analyse the data. More concretely, we applied an inductive content analysis process since our research was exploratory and the results had to be derived from the interviews. We used the coding technique to transform the qualitative data (interviews) into quantitative data (codes) following grounded theory coding principles [59]. The leading team performed four steps to identify the codes. First, they read independently the complete transcriptions and corroborate the consistency and coherence of the answers using those provided in question 6 for an individual finalized project. Each researcher selected the relevant text for each answer. Second, the researchers shared with each other the relevant text selected in order to agree on the information to be extracted. For the cases in which they could not reach such agreement, the researchers analyzed jointly the information again until reaching a consensus. Third, the researchers jointly defined a category or code for each relevant text extracted. They discussed the level of abstraction of the codes. And, finally, after the definition of all the codes, the researchers conducted a general revision where some codes were split, others were generalized and others were merged. The codified data is available online (see Footnote 5). After this categorization process, the leading team used frequency analysis to detect the recurrent cases reported in Section 5.

5 Results

In this section, we present the findings obtained from the analysis of the responses given by the interviewees. For each identified code (see Section 4.7), we include relevant quotations extracted from the interviewees' responses, and indicate the identifier of the company enclosed in parenthesis. These identifiers are two letter country codes according to ISO 3166-1 alpha-2 followed by the number of the participant (*e.g.*, "(PT1)" represents the first participant from Portugal). The name of the NFRs that the practitioners provided in the responses of the questionnaire were according to the predefined classification of NFRs in their companies, if any. When possible, we mapped those NFRs to the ones defined in ISO/IEC 25010 standard [24]. For each RQ we provide the most relevant findings (see Table 5 for a summary).

5.1 What Benefits can the Management of NFRs bring to Companies as part of their MDD Processes? (RQ1)

We first inquired about the importance of NFRs for companies adopting MDD (if NFRs were not significant, the rest of the survey would not be necessary). Then, we asked them about their perception of how well MDD approaches support specific NFRs. The perceptions are compared later, in Section 6.2, with the NFRs really supported when MDD approaches are finally adopted.

Finding 1. *While near half of the respondents consider that NFRs are as important as functional requirements, the same amount still consider functional requirements more important than NFRs.*

Seven out of eighteen respondents stated that NFRs are less relevant than functional requirements for projects

TABLE 5
Summary of Findings

RQ1	**What benefits can the management of NFRs bring to companies as part of their MDD processes?**
Finding 1	While near half of the respondents (7 out of 18) consider that NFRs are as important as functional requirements, the same amount still consider functional requirements more important than NFRs
Finding 2	When adopting MDD, the interviewees expect to achieve improvements in NFRs such as productivity, maintainability, and reusability. Other NFRs, such as obtaining high levels of performance or strong security needs, are perceived as harder to achieve when developing with MDD (see Figure 3)
RQ2	**How do MDD approaches adopted by companies support NFRs?**
Finding 3	More than half of the companies (11 out of 18) declared that their MDD approach requires adaptations to support the NFRs of the software produced
Finding 4	After MDD approaches are adopted by the companies, performance, maintainability, quality of code and productivity are the supported NFR types mentioned most often (see Figure 4)
Finding 5	The respondents reported three different stages to address NFRs: at modelling time, at code generation time and/or at testing time
Finding 6	The interviewees use modelling and transformation functions as techniques for supporting NFRs when using MDD (see Figure 5)
RQ3	**How do companies deal with NFRs when the adopted MDD approach does not support their management?**
Finding 7	Half of the respondents (9/18) declared that changing the code manually to satisfy NFRs is common practice, half did not (see Table 3)
Finding 8	When code is manually modified, the changes are *ad hoc* and maintenance is seriously compromised
Finding 9	The benefits of using MDD overcome the extra effort required to make manual adaptations to support NFRs. Even the use of MDD excluding NFRs still pays

that adopt MDD. Some reasons provided for this statement were: the customer's priorities (*e.g., "clients focus first on the functional requirements because the functionality is what initially matters"* (ES3)), the type of project (*e.g., "in all our modernization projects, the migrated application has to be systematically ISO-functional at least"* (FR3)) and modelling limitations (*"NFRs are not considered in modelling approaches"* (DE1)). However, the same amount of respondents considered both types of requirements equally important (*e.g., "at the end of the day, you have to fulfil all customer's requirements no matter if they are functional or not"* (ES1), *"quality is an intrinsic part of a successful industrial function"* (AT2)). Only one out of eighteen respondent stated that NFRs are more important than functional requirements *"because when some NFRs are not met, usually the functional requirements will need to be modified"* (DE3). Three respondents did not provide clear-cut answers.

Finding 2. *When adopting MDD, the interviewees expect to achieve improvements in NFRs such as productivity, maintainability, and reusability. Other NFRs, such as obtaining high levels of performance or strong security needs, are perceived as harder to achieve when developing with MDD.*

This finding provides insights on the expectations of respondents with respect to NFRs when applying MDD in their projects. Figure 3 shows that the majority of the interviewees believe that gains in productivity, maintainability, and reusability will be achieved when using MDD (*e.g., "Productivity definitely fits. You should be able to create an awesome application with a minimal number of steps"* (PT2), *"there are NFRs easier to meet with MDD, such as maintainability or reusability"* (ES2)) meaning that productivity, maintainability and reusability are NFRs expected to be satisfied and improved when MDD is adopted. In turn, performance and security are perceived as the most difficult ones to guarantee (*e.g., "time (performance) as manual optimization may be much better than the MDD result"* (AT2), *"others are harder to manage (such as security) because they are rather complex to model with the existing modelling languages and tools"* (ES1)). However, it is worth to remark that this response is not unanimous since some respondents did consider security and especially performance as expected NFRs achieved when adopting MDD. The reason behind this apparent contradiction is that they perceive that a basic level of performance and security using MDD may be satisfied (*e.g., "we can easily optimize code using model transformations"* (PT1), or *"security, for example, authentication, is always addressed automatically, an access system is guaranteed"* (PT3)). On the contrary, higher levels of performance or strong security needs are more difficult to ensure in MDD approaches (*e.g., "Performance as manual optimization may be much better than the MDD result"* (AT2)).

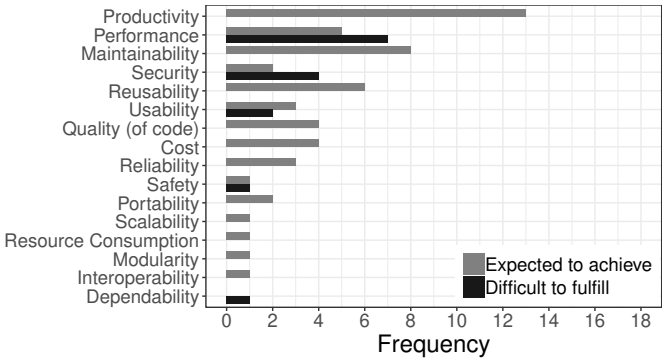


Fig. 3. NFRs expected to achieve and NFRs perceived as difficult to fulfill when applying MDD

The expected gains in productivity, maintainability, and reusability are mentioned by 14 out of 18 interviewees as one of the main factors that drove the adoption of MDD in their companies. Instead, we have not observed any link between the difficulties envisaged by the practitioners and the factors that discourage the MDD adoption.

We have noticed that from the practitioner's perception, NFRs improvements when MDD is adopted not only refer to NFRs related to the developed product but NFRs related to the development process. An example of this is that the most mentioned NFR is productivity, which is an NFR clearly tied to the MDD process itself rather than the developed product (see Section 6.1 for more details on this observation).

5.2 How do MDD Approaches Adopted by Companies Support NFRs? (RQ2)

While the former RQ was meant to address the perception of MDD as a method, this and the next RQs go deeper into the way in which companies currently use MDD in their projects to support NFRs. First, we asked whether the adopted MDD approach supports NFRs without adaptations. Next, we asked about the types of NFRs effectively supported and the stages in which NFRs are addressed. Then, we inquired about the different techniques used to embed NFRs in the MDD process.

Finding 3. *More than half of the companies declared that their MDD approach requires adaptations to support the NFRs of the software produced.*

Only five out of eighteen respondents use an MDD approach without adaptations to handle their NFRs (*e.g., "usability and security are addressed using the platform as it is"* (PT3), *"In one project related to game development, we could actually create a fully functioning game with relatively little effort. Games are, in general, probably, the pinnacle of usability. It definitely helped there"* (PT2)). Two out of eighteen interviewees did not respond to this question, because they do not correlate NFRs to MDD in general. Instead, eleven out of eigtheen companies reported adaptations in their MDD approaches to support or improve the NFRs of the software produced. The most common adaptations are:

- *New metamodels or extensions of existing metamodels* reported by seven interviewees. For example, *"new models/metamodels introduced (new kinds of models): new domain-model components, which need changes in the application model"* (AT2).
- *Specific technology adaptations* reported by four interviewees. For example, *"generators also have to be modified accordingly"* (FR3).
- *Definition or modification of transformations* reported by four interviewees. For example, *"New transformations: new code generation output needed, for example, user interfaces"* (AT2).

To illustrate individual adaptations, two companies reported adaptations made for a particular NFR (*e.g., "For portability we provide new transformations for each platform. The transformations are reusable for different projects"* (AT1), *"We implemented the Safety requirements by defining a set of model validations. We defined the appropriate constraints at the metamodel level, to ensure that the models we developed satisfied our Safety requirements. We also defined some validation rules for the code, to check that it satisfied the requirements and safety criteria imposed by the customer"* (ES1)).

Only five out of the eleven companies reporting adaptations mention that they are reusable (*e.g., "they can be generalized [...] whatever you model you can reuse the generic purpose"* (FR1)).

Finding 4. *After MDD approaches are adopted by the companies, performance, maintainability, quality of code and productivity are the supported NFR types mentioned most often.*

We asked for the types of NFRs supported by the MDD approaches adopted by companies. Figure 4 shows the results.

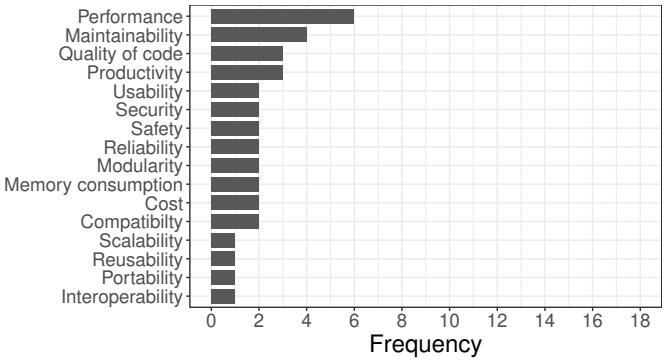


Fig. 4. NFRs Supported by the MDD Approaches Adopted by Companies

If we compare these results with Finding 2, we can observe that there are some NFR types which are better supported than expected. Remarkably, performance is considered by far the best supported NFR type while it was ranked fourth in terms of expectations. Conversely, reusability was expected to be supported by MDD approaches (ranked third in expectations) but in fact it was mentioned only by one respondent when it comes to perceived support.

It is worth mentioning that the level of achievement of a particular type of NFR depends on the company project or objectives, for example, a company may consider that an MDD approach can handle security because the level of security required is basic. In fact, some respondents argued that although MDD directly treats some NFRs, if they want to improve them, they have to make some adaptations like, for instance, to improve code generators. This is the case of performance. Although high levels of performance are difficult to achieve using MDD, acceptable levels may be obtained adapting MDD approaches (*e.g., "We have improved the code generator to reuse better the code and improve performance"* (FR1)).

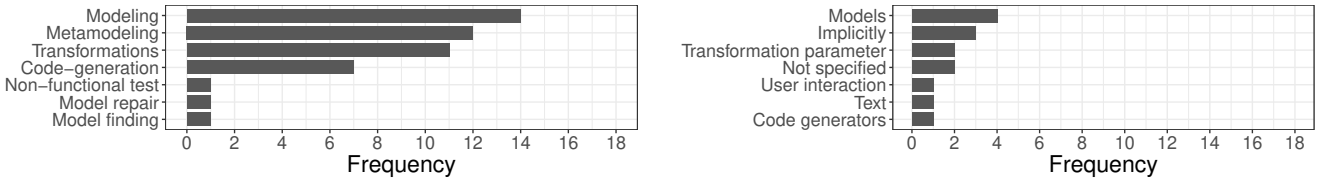
Finding 5. *The respondents reported three different stages to address NFRs: at modelling time, at code generation time and/or at testing time.*

The best way to deal with NFRs would be *"to inject NFR data into the handled models"* (FR2) and from them automatically generate the resulting code. Unfortunately, for some companies *"the current way to do it is by human interaction[6] or evaluation"* (FR2).

Fourteen out of eighteen respondents reported that they were able to include some types of NFRs in the models at the initial stages of development (*e.g., "Models are used to specify usability; for example, a base system layout is used and modified by us"* (PT3)). They agreed that considering NFRs from the beginning of the MDD process facilitates their fulfilment (*e.g., "a bad modelling [of NFRs] at the beginning can have a substantial impact on all the rest of the software development and also on the finally produced software"* (FR2)).

Other types of NFRs are more difficult to specify through models and are tackled during development, as it is reported by eleven out of eighteen respondents. In some cases,

6. The respondent meant "interactively", i.e. human working with a tool.



(a) Techniques (b) Specification Mechanisms

Fig. 5. Techniques and Specification Mechanisms to Support NFRs in MDD

the MDD process itself is tailored to support specific NFRs (*e.g., "if some NFRs are strongly expressed by customers, it may have an impact on the deployed MDD process"* (FR3)).

Finally, one out of eighteen respondents mentioned that particular types of NFRs are not really specified but validated or monitored at the final stages of development (*e.g., "we have some non-functional tests to see if performance degrades when we do something"* (FR1)).

Finding 6. *The interviewees use modelling and transformation functions as techniques for supporting NFRs when using MDD.*

Figure 5a shows that (meta-)modelling, transformations and code generation are the main techniques for supporting NFRs. Besides, when NFRs are specified, that is, provided as an input to the MDD process, they are usually specified in models (see Figure 5b). According to the responses, the companies use, mainly, the following techniques to support NFRs:

- *Modelling or metamodelling.* NFRs are modelled through explicit constructs. For example, usability (using *"[user] interface models - a base system layout is used and adapted by us"* (PT3)), memory consumption (through *"model annotations"* (AT3)) and performance *"specified through profiles"* (IT2). In three cases, the respondents mentioned that they represent NFRs only in UML models *"extended with stereotypes and tagged values"* (AT3).
- *Transformation functions.* In this case, NFRs become transformations or transformation parameters. *"We use [...] transformations (model to code, but not model to model yet), domain-specific languages and language families"* (PT1), *"by transformation (as a transformation parameter), and through the MDD process. I think it is a bit of both"* (PT2).

5.3 How do Companies Deal with NFRs when the Adopted MDD Approach does not Support their Management? (RQ3)

In the third and last RQ, we wanted to know the practices of companies when the MDD process does not completely handle the NFRs that apply to the system under development. We also wanted to explore their opinions on the impact of such *ad hoc* management on the full process.

Finding 7. *Half of the respondents declared that changing the code manually to satisfy NFRs is common practice, half did not.*

Table 6 shows the practices of companies for dealing with NFRs out of the MDD process. Two out of eighteen respondents did not respond to this question, because they *"cannot correlate the NFRs to MDD in general"* (AT1) or because no code was generated from the models.

Nine out of eighteen interviewees expressed that *"manual modification is needed either on the code or the original models [to support specific NFRs]"* (FR2). In more detail, eight out of the nine respondents reported changes in the generated code, even at the cost of losing some of the reusability and maintenance brought by MDD. The remaining respondent, PT1, declared that the code is only extended but not modified, in order not to compromise the MDD process. On the contrary, the other seven interviewees stated that *"[we] do not change what was generated. We never tweak generated code"* (PT2) and that *"we prefer to incorporate at the model/transformation level whatever we need"* (ES1).

Eight out of the nine respondents reporting manual changes in the generated code provided more details for specific types of NFRs. Two concrete examples were:

- *Security. "Generated code cannot always be proven as 100% safe or secure and some customer may still trust more full human developments for critical parts of the code"* (FR2))
- *Performance. "Time [performance] as manual optimization may be much better than the MDD result"* (AT3).

Finding 8. *When code is manually modified, the changes are ad hoc and maintenance is seriously compromised.*

When manually modifying the code, the most common approach is to perform *"manual changes in specific parts of the generated code"* (AT2). Seven out of the nine respondents that modify the resulting code follow this approach. Only one of the seven previous respondents mentioned that the manual modifications were persistent across MDD iterations (*e.g., "we use the concept of protected areas that are manually developed portions of the code that are preserved in case of code regeneration"* (IT3)). The other remaining companies (two out of the nine respondents) modify code *"just to complete what was generated"* (PT1) and *"to rewrite the bad code, but this is violating some reusability and maintenance NFRs brought by code generation"* (FR2).

Finding 9. *The benefits of using MDD overcome the extra effort required to make manual adaptations to support NFRs. Even the use of MDD excluding NFRs still pays.*

When we inquired if MDD adaptations would pay off (*e.g.,* adding or customizing some model transformation), we appreciated positive responses from the eighteen respondents (*e.g., "even if you have to do some adaptations [to support NFRs], I think the extra effort is worth it"* (FR2)). It is

TABLE 6
Practices for Dealing with NFRs out of the MDD Process

Description	*Representative quotes*	*Freq.*
The generated code can be changed.	*"Partial modification of the code produced. Manual changes in specific parts of the generated code (framework)"* (AT2) *"Hard real-time requirements could not be implemented with the models"* (DE3) *"In a few cases, it has to be modified for sake of performance"* (IT1)	4
The generated code can be changed, although it is preferable to change the code generator.	*"For example, when the resulting application does not provide the required performance. Thus, you may have to refactor the end-user application, that is, change the generated code. This case has happened to us. What we have in our company is a procedure for reporting this kind of issues, so that the generator can be modified as soon as possible. Our generators are subject, like any other software we develop, to maintenance. With this, the answer is that we can do it in exceptional circumstances, but in a controlled and documented manner"* (ES3)	4
No changes in the generated code, but extensions used to add new code.	*"We do not modify the generated code, but we do extend it. The intention of the model is totally fulfilled, so we generate all we want to generate at this moment. The extensions we then provide are things not captured by the model"* (PT1)	1
No changes in the generated code.	*"The part that is generated is never modified by hand"* (FR1) *"No, the generated code is not modified"* (PT3)	7
Not applicable.	*"Question not relevant as the approach applied is different to the scenarios provided"* (DE1)	2

also seen as a trade-off: the inherent benefits of MDD compensate this extra effort (*i.e.,* increase in maintainability and improved reusability). Some specific scenarios may support this opinion, *e.g., "especially when you are going to generate code for different platforms or systems, or for different versions of the product"* (ES1). One of the eighteen respondents mentioned that the adoption of MDD itself is the costly part, *"once [...] adopted, then people are well-aware that they can easily benefit of [MDD approaches] also in the field of NFRs"* (IT1).

When we asked whether manual modifications pay off (*e.g.,* changes in the produced code), all responses (eight out of the nine reporting manual modifications) were positive except for one. The positive answers were mostly of pragmatic nature (*e.g., "the modification effort is worth it as you can capitalize on the improvements"* (FR2)). The interviewee reasoning against said: *"when you have a lot of artefacts generated from your MDD approach (...) you can stop worrying about some technical details"* (PT2), meaning that for this interviewee the primary benefit of MDD is to avoid dealing with technical aspects and performing manual modifications *"defeats the purpose"* (PT2).

Finally, when we asked whether the use of MDD excluding some NFRs still pays off, only five responses out of eighteen were obtained, four of them were still positive when *"you just exclude a couple of NFRs, and all others are supported"* (FR3).

6 Discussion

In the following, we discuss our results structured according to the research questions.

6.1 What Benefits can the Management of NFRs bring to Companies as part of their MDD Processes? (RQ1)

Answer RQ1. *The adoption of MDD does not impact on the perception of importance of NFR during the development process. In MDD, NFRs are considered less or equally important than functional requirements. Improvements in productivity, maintainability, and reusability are expected to be achieved.*

A consistent finding of the survey is: NFRs are not perceived as more important than functional requirements in MDD processes (Finding 1). Other studies, not specifically linked to MDD, even suggest that NFRs, especially those difficult to elicit (such as security-related ones) are considered less important than functionality and, often, they are retrofitted or pursued in parallel with (but separately from) functional requirements [60]. Berntsson Svensson *et al.* [40] offer two reasons for this imbalance:

- *NFRs have lower priority than functional requirements.* The reason argued by the respondents is that NFRs, such those related to usability, are not considered in the early stages of development. This fact is supported by our study results. As reported in Finding 5, the NFRs that are difficult to specify through models are often handled by modifying the generated code or even validated during the final stages.
- *NFRs are more difficult to discover.* The main problem with NFRs is that many of them remain undiscovered or in non-measurable terms. This opinion is also given by software architects [28] who declare NFRs as difficult to elicit upfront. Recent approaches suggest using data-driven methods to facilitate this elicitation [61] if the system already produces data. New proposals capable of automatically generating NFRs, as the one presented in [62], may change the perception of the importance of NFRs.

Organizations participating in a study by Borg *et al.* [54] stated that their main focus is on functional requirements, primarily because existing methods focus on them. This argumentation applies well to the MDD case. As reported in our study, current MDD approaches hardly consider NFRs. Therefore, adopters of these approaches may tend to focus on functionality because they are not capable of handling NFRs. Thus, we may speculate that the production of improved MDD methods, which include NFRs in the models and transformations, could positively influence the importance given to NFRs.

We have only found one study reporting the majority of the respondents to consider NFRs and functionalities at the same level of importance [50]. Interestingly, they found a statistically significant relationship between the importance given to NFRs and their implicit or explicit nature in the project. More precisely, the study revealed a

high probability that projects, which consider functionality and NFRs at the same level of importance, also treat NFRs explicitly. This correlation is in tune with the findings in our study. As reported in Finding 1, when analysing the 15 responses (3 did not answer) given by the companies, we found that 8 companies consider NFRs equally or more important than functionality. We observed that 7 out of those 8 companies adapt their MDD approach in order to take NFRs into account, thus, making them explicit. However, 4 out of 7 companies that state NFRs to be less important than functionalities use MDD without any adaptation. These responses, if further corroborated, may indicate that giving high importance to NFRs induces the adaptation of the MDD approach or that the capability to adapt the MDD approach increases the awareness of the importance of the NFRs.

When looking into other factors that may have affected the perception of NFRs' importance, we observed that 3 out of 4 companies belonging to the embedded systems domain claim that all types of requirements are equally important. One reason for this may be the criticality of some quality aspects in this kind of systems, such as the ones for aeronautical and defence systems, with respect to final acceptance and compliance with standards (*e.g.*, safety standards). In fact, in some of these systems is common to follow a quality certification process (*e.g.*, CMMI[7]) which implies dealing with some types of NFRs. A consequence of this observation may be the convenience of defining domain-dependent MDD approaches, in which critical domains, such as forensic systems, may include NFRs at the heart of their methods, while other domains could still primary focus on functionalities.

Although NFRs are not considered more important than functional requirements for the respondents of our study, some NFRs are expected to be supported when MDD is adopted and are considered key factors for the adoption of MDD. We refer to productivity, maintainability and reusability, which were the most mentioned in Finding 2. Remarkably, practitioners' expectations are more focused on NFRs related to the development process than on NFRs related to the software product. This fact is aligned to what we found in the literature (*e.g., "MDD can shorten the development of complex solutions"* [41], *"applying MDE increases productivity and shorten development time"* [51]). Liebel *et al.* [49] found out that 68% of the participants who took part in their study stated that their companies adopted Model-Based Engineering (MBE) approaches because they needed shorter development time and to improve reusability. This is in tune with our observations leading to Finding 2. As reported in Section 5.1, 77% (14 out of 18) of the interviewees said that productivity, maintainability and reusability improvements are one of the main reasons for adopting MBE.

If we examine the three NFR types reported in Finding 2, namely productivity, maintainability and reusability, we may notice that all of them are of internal nature (perceived by technical staff when they are applying the MDD process). In consequence, since final users will not require these internal NFRs, they are harder to elicit and may remain implicit. Moreover, these NFRs can be handled with the techniques and tools provided by MDD (*"We have improved the code generator to reuse the code better and to improve performance"* (FR1); *"Maintenance is also made easier via code generation, as multiple errors can be detected, and the code generators updated accordingly to fix them"* (FR2)).

In turn, NFRs, such as performance and security, are perceived as harder to cope with when MDD is adopted (Finding 2). These NFRs are explicit requirements and of external nature (required and perceived by the final users), *e.g.*, final users may notice different levels of performance of a software product. Although MDD techniques and tools provide support for these NFR types (for example, PT3's MDD approaches support security operationalizations and PT1 MDD transformations support some levels of code optimization), complex levels of performance and security are perceived as difficult to achieve and require, in most cases, manual modifications (*"I would say that code quality is quite well supported via code generation, some security and performance aspects too, to some extent"* (FR2); *"Time (performance) as manual optimization may be much better than the MDD result"* (AT3)). When we analyse, in our study, the companies reporting performance or security as NFRs perceived as difficult to guarantee in MDD, we observe, in general, large companies, with a high level of MDD adoption (L3-L4, L4 and L5) and using MDD with adaptations. This means that performing adaptations and having a high level of MDD adoption do not ensure that companies have confidence in performing a good treatment of complex NFRs (as it is the case of AT2, DE3, ES1, ES3, FR1, FR2 and PT2). Although one could call for improved MDD approaches handling better these particular NFR types, we think that this current limitation may, in fact, represent a boundary of MDD as a productive development method. In other words, we think that complex requirements such as NFRs, which have fundamentally a cross-cutting nature, are difficult to quantify, and are highly interrelated with each other, may be difficult to handle systematically and may require a specific treatment. To this respect, it would be optimal that the code written manually to meet these complex NFRs is clearly separated from the code generated by the chosen MDD approach, *e.g.*, as reported by one of our interviewees (IT3) with the concept of "protected area" or by implementing the Open-Closed Principle [63] in the solution, allowing modifications to be persistent across MDD iterations.

As a closing important remark, it is worth to mention that the scarce support for NFRs in MDD processes may be a consequence of the way in which companies manage NFRs. If companies do not properly elicit or document NFRs beforehand, they can hardly bring them into the MDD process. For instance, 10 out of the 18 respondents reported that they do not use any type of classification schema for NFRs, instead *"NFRs are specified case-by-case via the projects requirements specifications"* (FR3). The lack of adoption of classification schemas is one indicator about the ad hoc treatment of NFRs by companies. Therefore, it might be necessary to increase the awareness and capacities of companies on the importance of NFRs before any other related further action. Current works in data-driven elicitation of NFRs [62] or systematic ways to document NFRs in agile software development backlogs [64] may definitively help in this direction.

7. https://cmmiinstitute.com

6.2 How do MDD Approaches Adopted by Companies Support NFRs? (RQ2)

Answer RQ2. *MDD approaches that support NFRs are capable of only dealing with specific types of NFRs. Performance, maintenance, quality of code and productivity are the most mentioned types. NFRs are either handled implicitly (*e.g.*, as part of model transformations) or retrofitted explicitly at the end of the transformation chain (*e.g.*, by modifying the generated code).*

Adopting a particular MDD approach may work well for specific types of NFRs but in general the situation is not optimal. Comparing the results reported in Figure 3 (dark coloured bar) and Figure 4, our study shows that the support of current MDD approaches to NFR types is worse than the expectations that practitioners had. This observation can be considered as an indicator that NFRs are still not managed satisfactorily by current MDD approaches. In fact, when handling certain NFR types or many types at once, things get complicated and the MDD process needs to be adapted (Finding 3).

There are two perspectives in literature to address NFRs: 1) the use of MDD yields satisfaction of some types of NFRs even without adapting the MDD process; and, 2) companies are using MDD adaptations to address NFRs specifically. For the second case, Ameller *et al.* [8] mentioned different stages of the MDD process to address the NFRs: at modelling time and at code generation time. However, in this study, participants also said to address NFRs at testing time (Finding 5).

The first perspective refers to emergent or inherent properties obtained by using MDD without adaptations, such as gains in maintainability, productivity and quality of code. As discussed in Finding 4, those NFRs are better supported by MDD approaches. For example, Büttner *et al.* [65] showed a reduced number of bugs thanks to the quality control performed at model level and the adequate transformations performed from the specifications. Moreover, they pointed out a productivity improvement due to the separation of higher-level concerns from implementation details. This is in line with what we found in our study. Other authors have also reported significant gains in productivity and maintenance when adopting MDD (*e.g.*, [38], [49], [53]).

The second perspective refers to adapting MDD to handle particular NFRs. These adaptations may include new models, metamodels, UML profiles, code generators and new transformations [8], as also discussed in Finding 6. There are examples of those adaptations (*e.g.*, using model transformations) to incorporate NFRs such as performance [51]. However, these adaptations require some investment, which seems to pay off only if reusability is possible. Assessing the kind of reusability (global, within an application domain, within a set of projects, across technological platforms) against its cost is an open question.

Some researchers question the gain on performance when MDD approaches are used. The reasons behind this position are diverse. Specifically, many stakeholders do not have a good understanding of what level of performance is required to support their business processes [66], while MDD approaches require an accurate understanding of performance to explicitly capture such constraints in the created models. Moreover, performance requirements are more often dismissed due to the difficulties in estimation [40] and in cases they are considered, the performance of the generated code meant to be poor [49].

There are several proposals to include performance in MDD, particularly to predict the performance of the generated software (*e.g.*, [33], [67]). If the MDD research community finds a way to convince practitioners of the relevance of their proposals, this could significantly raise the value of MDD as a valid tool to ensure the satisfaction of this NFR. However, this is an already reported problem [68].

The current state of MDD technologies as well as the difficulties in handling NFRs in a seamless manner still pose many challenges to other types of NFRs. It is commonly accepted that there is no universal approach for the NFR-specific elicitation, documentation, and analysis. NFRs are usually described vaguely [28], [54] and are often not quantified [40]. More generally, many functional requirements are even misclassified as NFRs, *i.e.*, there seems to be a general misconception in practice (and sometimes in research as well) about many NFRs as they appear in the wild [69]. The outcome results are difficult to analyse and test [28], [40], [54]. Additionally, as NFRs are often retrofitted in the development process [20], they are often implicitly managed with little or no consequence analysis [40]. Overall, this suggests that several of these NFRs are not yet so well understood, which naturally renders the development of adequate MDD tools for supporting them difficult. While we can find niches for each of these NFRs where there are MDD solutions, they are very broad in nature, which helps explaining why their automation is still challenging.

Adapting the MDD process, creating new models, metamodels, and new transformation techniques is still a task that requires investment. New metamodels and other artefacts need to be defined to manage some NFRs in MDD approaches. In order to make this investment pay off, reusability is a key factor. However, further studies are required to know more about reusability, *e.g.*, about the factors that make reusability possible. More specifically, we need to better understand if the reusable adaptations are specific to a project, specific to a product line or to a concrete domain, or on the contrary, they are generic enough to be reused in a wide spectrum of projects. In line with our opinion in Section 6.1, not only reusability is a key factor for having a return on investment, but also understanding what NFR types can be effectively managed better in the MDD process with these improvements. The results gathered in this study about supported NFR types are useful at this respect: we can argue that it is better to invest effort in those types as performance and maintainability (see Figure 4) that have been reported as the best supported by current MDD approaches. Of course, the counter-argument could be to make an effort in improving the current situation of types as portability or interoperability which are not well-supported according to our study. We could think in fact that this second type of NFR types could be attractive to researchers willing to open new research challenges in the field.

6.3 How do Companies Deal with NFRs when the Adopted MDD Approach does not Support their Management? (RQ3)

Answer RQ3. *Modifying the generated code seems to be the only viable solution when NFRs are not directly supported in the MDD process. However, a significant percentage of MDD engineers prefer to avoid such modifications even at the cost of not being fully compliant with the requirements specification. Even in these cases, the general opinion is that MDD still pays off.*

As discussed in Finding 3, eleven of the interviewed companies reported adaptations (*e.g.,* new metamodels or extension of existing ones, or new transformations) in their MDD processes to support some required NFRs. On the contrary, half of our interviewees reported manual code changes to deal with other NFRs (Finding 7).

It is generally agreed that in MDD, any type of change should be made only at the modelling level and then re-generate the code, rather than modifying the generated code, in order to avoid losing the changes made. Hutchinson *et al.* [38] reported that almost 70% of their respondents mainly made modifications to their models, whilst around 15% did not. In contrast to that finding, our study points out that for the specific case of unsupported types of NFRs, half of the companies deal with them by directly changing or rewriting the resulting code or by adding new code. The reason behind this contradiction may be that some types of NFRs do not have full support neither at modelling level nor during the code generation process. Thus, practitioners are required to find *ad hoc* solutions implementing NFRs directly in the code increasing the software development cost and even at the cost of losing reusability and maintenance capabilities. However, current modelling languages are able to describe most functional requirements and are able to generate the code without further modifications. This fact may suggest the need for companies to recruit qualified workers able to create or customize MDD techniques or languages for modelling and managing NFRs, which may become one main challenge to achieve such goal.

In case of code modification or extension, our study reveals that changes in the resulting code are rarely kept consistent with the source model. Therefore, changes are lost in case of code re-generation. This result is in tune with other works [44], [70], both pointing to code modification as a source of inconsistency and lack of persistence.

In this sense, as a first step to prevent losing manual modifications, there are tool-supported solutions such as Acceleo[8] or Xtend[9] supporting a way of specifying the "protected areas" in the generated code as also mentioned by one of our interviewees (IT3). User code blocks delimit portions of text that are only generated once and preserved for subsequent re-generations. Although such extensions can be an important feature to MDD tools, we argue that they are still insufficient to guarantee consistency between code and models: traceability is also needed for supporting round-trip engineering [44]. Clearly, this important issue requires more research and practical (tool supported) solutions.

8. https://www.eclipse.org/acceleo
9. https://www.eclipse.org/xtend

We have found a quite uniform consensus among our interviewees that adapting MDD pays off despite the extra effort required (Finding 9). We only found one exception to this positive attitude: when the number of non-supported types of NFRs largely exceeds the supported ones (PT1). Those that consider that MDD pays off is especially because it represents a long-term investment: once the effort has been invested, several NFRs (such as maintainability- and portability-related ones) are easier to manage. Another possible explanation, as suggested by Ameller *et al.* [28], is that even if MDD does not support some types of NFRs, other (non-MDD) methods may also have difficulties dealing with NFRs.

7 Threats to Validity

Threats to validity are inherent to every empirical study. We discuss the threats as suggested by seminal publications in empirical software engineering (*e.g.,* [9], [57], [71]) and, where reasonable and possible, the mitigation actions we employed. We classify the threats into internal, construct, external and conclusion validity [71].

7.1 Internal Validity

Communication. 1) The interviewees may have understood some questions differently, resulting in lower quality or even invalid answers. *Mitigation.* We designed the questionnaire following the indications that are given by Dillman *et al.* [57] and we piloted it in two iterations to ensure its understandability. 2) We wrote the questionnaire in English, which is not the native language of all the participants. *Mitigation.* We allowed participants to decide upon their preferred language during the interviews. Hence, it was necessary to translate the transcription of the interviews to English in those cases where the interviewees preferred to use their native language. Related to this process, some information may have been lost (*e.g.,* in those three cases where only one researcher served as interviewer), or not translated properly (*e.g.,* the technical terms). *Mitigation.* While we do not see major differences in the quality of the transcripts themselves, as all were based on recordings and most interviews (15 out of 18) were conducted by multiple researchers, we still tried to reduce the threats by local researchers verifying the contents of each finally translated summarising transcripts in an additional validation step.

Background. 1) Respondents may not have had the proper knowledge profile as to provide useful answers. *Mitigation.* We selected only practitioners with sufficient experience using MDD for the development of software products in their current company. 2) Even though participants are using MDD, their understanding of MDD (and of NFRs) might still differ from the common understanding of the concepts and principles incorporated by MDD. *Mitigation.* We incorporated specific questions and scenarios to uncover their understanding about MDD.

Interviewees bias. 1) Participants may have been reluctant to provide sincere responses when explaining the negative aspects about their work and their company. *Mitigation.* We highlighted the anonymity of the respondents and their companies in the published material, and we clarified that

the purpose of the interview was not to evaluate or assess their current situation or even their project environment and company. 2) Our respondents may have been slightly biased towards accepting new technological trends such as MDD. In contrast, other roles such as managers tend to be more conservative when adopting new technologies. This may be reflected in some of the answers obtained. However, in many cases, the participants were both technology experts and managers (*e.g.,* project leader). Therefore, we consider that this particular bias should be substantially reduced.

Interviewers bias. Due to the context of this study, the interviewers themselves are also a source of bias, because several different interviewers executed the interviews. Hence, there is a risk that the interviewers might have conducted the interviews in different ways (*e.g.,* by putting more emphasis on specific parts of the questionnaire than on others). *Mitigation.* We documented the material used for this survey, including a specific guide for conducting the interview (see Section 4.5) that provided instructions for all the interviewers of the study. Moreover, the interviewers contacted on regular basis with each other.

7.2 Construct Validity

Protocol. Concerning the overall underlying methodology, we can never assure that the protocol designed in advance is complete and that it provides sufficient details for the success of a study. *Mitigation.* We followed Ciolkowski *et al.'s* guidelines [9] and consolidated the final protocol (see Footnote 2) as part of a validation phase where all involved researchers checked the first version of the protocol proposed by the leading team. Furthermore, the protocol has further undergone external peer-review as we published and presented it at the IEEE International Requirements Engineering Conference in 2015 [55].

Questionnaire. 1) The resulting questionnaire might not sufficiently cover our research questions. Missing relevant data during the interviews could in general lead to the impossibility to provide an answer to some of the research questions. *Mitigation.* We mapped each question in the questionnaire to one or more variables (to be used for the analysis), and these variables in turn with the research questions (one variable may provide insightful information for more than one research question). Furthermore, we carefully designed a conceptual model (available in the protocol) that clarifies the relationships between the variables (*e.g.,* context, input and output factors of the MDD process), which additionally strengthens our confidence in the accuracy and sufficient completeness of our questionnaire. 2) We performed an exhaustive consistency verification of the obtained data and, in fact, we had to deal with some inconsistent data during the analysis (*e.g.,* one interviewee stated that no adaptations were done to the MDD process, but afterwards he mentioned some modifications to improve the efficiency). *Mitigation.* We actively approached the participants to validate the data during the analysis to resolve such issues.

7.3 External Validity

As it is always the case in interview-based surveys, the size of the sample and the sampling technique used do not provide the statistical basis to generalize the results to the target population. Generalizability is further threatened by the incompleteness of some interviews that did not provide answers to all the questions included in the protocol, making difficult even generalization by analogy. However, even the descriptive analysis of the results we obtained interesting insight that was not available before. Furthermore, we mitigated this threat as much as we could by carefully characterizing the context of the participating companies.

7.4 Conclusion Validity

Considering the threats described before, one question that arises concerns the degree to which the conclusions we draw from our data are reasonable. Given the qualitative nature of our study in light of the corresponding mitigation actions we employed to arising threats, we are confident that the conclusions drawn are accurate and reasonable to the extent qualitative studies allow, that we achieved our overall goal to provide an in-depth analysis of the state of the practice to support problem-driven research in our field, and that the challenges identified by our results, aligned with the context factors, can be used to steer future research for specific settings.

However, to build a more robust theory, we should triangulate this research with further studies relying on other data sources and eventually using other empirical methods. These studies could be designed on top of the protocol and the questionnaire of the survey reported in this paper. Furthermore, we see the value of replicating empirical studies. Therefore, we made all related material available (see footnotes 2, 3, 4 and 5) under the CC-BY license[10]. The open nature of our project shall support researchers and practitioners to replicate this study and, in the long run, to evolve the results and generalize them based on a more global understanding of the context.

8 Conclusions

In this paper, we have presented the results of a family of interviews in eighteen European companies from six different countries that use MDD in some or all of their software projects to better understand how NFRs are handled when developing under the banner of MDD. During the analysis of our data, we identified nine major findings (see Section 5) and used those to answer to our research questions in relation to existing evidence (see Section 6). One hope we associate with our work is to not only directly contribute to the existing body of knowledge of handling NFRs in context of MDD – which currently is still weak – but also to encourage other researchers to join in exploring this highly important topic area.

Looking at the results of this study, we can classify the management of NFRs in MDD processes into three categories: a) natively, where the MDD method is able to cope with (some selected types of) NFRs; b) through extensions, where the company adds or changes the languages or transformations to handle NFRs; c) ad hoc, where companies can modify the source code as they need. In most cases we

10. https://creativecommons.org/licenses/by/4.0

have seen only partial support, and only for very specific types of NFRs. Even if there are different positions among the interviewees, modifying the generated code to support NFRs seems to be the most widely used solution so far. This approach may be suitable in some scenarios, such as building software prototypes. However, software products need to be compliant with all types of requirements and MDD seems to be not ready for such diversity of needs. In particular, the study has shown that software engineers expect MDD bringing benefits for some particular types of NFRs related to the development process, while companies adopt MDD approaches mainly to better support NFRs related to the system behaviour.

Retrofitting NFRs in MDD after the fact is in tune with the state of the practice in rather traditional engineering approaches, but it further has major impacts on inconsistencies when regenerations cause modifications to get lost. Therefore, new approaches and techniques need to be proposed by the research community to improve the NFR support in MDD and solve the problems enumerated above. Otherwise, NFRs remain a delicate by-product casually attached at the very end of the engineering cycle. In addition, mismatches between industry practices and MDD teaching should be solved. As reported in [72], (1) a significant number of successful MDD companies build their own modelling languages and generators, suggesting a re-orientation of education away from UML notation to fundamental modelling principles, (2) MDD is generally taught top-down, whereas industry success is more likely when MDD is applied bottom-up and (3) successful application of MDD requires skills both in abstract modelling and compilers/optimization; however, these skills tend to be separated in standard computer science curricula.

Based on our observations, we propose the following research agenda:

- *Empirical studies.* In this study we covered six different countries from Europe, but we need to replicate it in other parts of the world (*e.g.,* North America or Asia) to get a broader view on the current state of the practice. In particular, NFRs are not being studied enough in the context of MDD. More empirical studies are needed to further strengthen our initial observations. Despite the replications, we postulate the need for further inquiry methods based on our first indicators revealed in our manuscript. Additional (and more diverse) data would help us as well to observe whether different geographical regions follow divergent practices regarding the handling of NFRs in MDD processes.
- *New approaches to handle NFRs in MDD.* Looking at the current practice, it appears that the MDD community has failed to identify a common suitable approach to handle NFRs in MDD. Researchers have proposed several solutions in the literature, but none of them seems to have been able to reach the maturity level required to be used in industry. Immature tools are still one of the main barriers for MDD adoption.

Finally, we share a quote from one of our interviews as it best describes a pragmatic view on the current state of MDD: *"The customer does not care if MDD or other technologies are used. He wants you to have a product that works as he wants, in time, within budget and with high quality. We are noticing that although MDD was cool a few years ago and the use of models sold, now customers no longer perceive it as a good solution. Many of them have had experiences that have shown them that this technology is not as good as we told them it was, and that it has the same problems as the rest of the technologies we have been trying to sell them for many years to solve their problems, without success"* (ES3). Improving the way of handling NFRs in MDD can be one of the several actions designed to revert this opinion.

ACKNOWLEDGMENTS

The authors thank Richard Berntsson Svensson, Maya Daneva, Grischa Liebel, and Bernhard Schätz for their contributions to this work as well as all participating industry partners for dedicating their time and for contributing with their experiences to our study.

This work was supported in part by a grant from *NOVA LINCS Research Laboratory* (Ref. UID/CEC/04516/2013), by the Spanish projects TIN2016-79269-R and TIN2014-52034-R, by the Austrian *Federal Ministry for Digital, Business and Enterprise* and the *National Foundation for Research, Technology and Development*, and by an ECSEL (*Electronic Component Systems for European Leadership Joint Undertaking*) project named *MegaM@Rt2* (grant agreement No 737494).

In memory of Bernhard Schätz, an inspiring scientist and a wonderful friend.

REFERENCES

[1] C. Atkinson and T. Kuhne, "Model-Driven Development: a metamodeling foundation," *IEEE Software*, vol. 20, no. 5, pp. 36–41, 2003.

[2] S. J. Mellor, A. N. Clark, and T. Futagami, "Guest Editors' Introduction: Model-Driven Development," *IEEE Software*, vol. 20, pp. 14–18, 2003.

[3] J. Whittle, J. Hutchinson, and M. Rouncefield, "The State of Practice in Model-Driven Engineering," *IEEE Software*, vol. 31, no. 3, pp. 79–85, 2014.

[4] P. Mohagheghi, W. Gilani, A. Stefanescu, and M. A. Fernández, "An empirical study of the state of the practice and acceptance of Model-Driven Engineering in four industrial cases," *Empirical Software Engineering*, vol. 18, no. 1, pp. 89–116, 2013.

[5] D. D. Ruscio, R. F. Paige, and A. Pierantonio, "Guest editorial to the special issue on Success Stories in Model Driven Engineering," *Science of Computer Programming*, vol. 89, pp. 69–70, 2014.

[6] J. Whittle, J. Hutchinson, M. Rouncefield, H. Burden, and R. Heldal, "Industrial Adoption of Model-Driven Engineering: Are the To ols Really the Problem?" in *Proc. of the 16th International Conference on Model-Driven Engineering Languages and Systems*. Springer Berlin Heidelberg, 2013, pp. 1–17.

[7] D. Ameller, X. Burgués, O. Collell, D. Costal, X. Franch, and M. P. Papazoglou, "Development of Service-Oriented Architectures using Model-Driven Development: A mapping study," *Information and Software Technology*, vol. 62, pp. 42–66, 2015.

[8] D. Ameller, X. Franch, and J. Cabot, "Dealing with Non-Functional Requirements in Model-Driven Development," in *Proc. of the 18th IEEE International Requirements Engineering Conference*, 2010, pp. 189–198.

[9] M. Ciolkowski, O. Laitenberger, S. Vegas, and S. Biffl, *Practical Experiences in the Design and Conduct of Surveys in Empirical Software Engineering*. Springer Berlin Heidelberg, 2003, pp. 104–128.

[10] M. Brambilla, J. Cabot, and M. Wimmer, "Model-Driven Software Engineering in Practice, Second Edition," *Synthesis Lectures on Software Engineering*, vol. 3, no. 1, pp. 1–207, 2017.

[11] Object Management Group (OMG), "MDA Guide rev. 2.0," Tech. Rep., 2014. [Online]. Available: http://www.omg.org/cgi-bin/doc?ormsc/14-06-01

[12] J. Bezivin and O. Gerbe, "Towards a precise definition of the OMG/MDA framework," in *Proc. of the 16th Annual International Conference on Automated Software Engineering*, 2001, pp. 273–280.

[13] F. Jouault and J. Bézivin, "KM3: A DSL for Metamodel Specification," in *Proc. of the 8th International Conference on Formal Methods for Open Object-Based Distributed Systems*. Springer Berlin Heidelberg, 2006, pp. 171–185.

[14] C. Atkinson, T. Kühne, and J. de Lara, "Editorial to the theme issue on multi-level modeling," *Software and System Modeling*, vol. 17, no. 1, pp. 163–165, 2018. [Online]. Available: https://doi.org/10.1007/s10270-016-0565-6

[15] A. Nugroho and M. R. V. Chaudron, *Evaluating the Impact of UML Modeling on Software Quality: An Industrial Case Study*. Springer Berlin Heidelberg, 2009, pp. 181–195.

[16] Object Management Group (OMG), "Unified Modeling Language (UML), V2.4 – Superstructure specification," Tech. Rep., 2014. [Online]. Available: http://www.omg.org/spec/UML/2.4/Superstructure/PDF

[17] M. Mernik, J. Heering, and A. M. Sloane, "When and How to Develop Domain-specific Languages," *ACM Comput. Surv.*, vol. 37, no. 4, pp. 316–344, 2005.

[18] M. Glinz, "On non-functional requirements," in *Requirements Engineering Conference, 2007. RE'07. 15th IEEE International*. IEEE, 2007, pp. 21–26.

[19] G. Kotonya and I. Sommerville, *Requirements engineering: processes and techniques*. Wiley Publishing, 1998.

[20] L. Chung and J. C. S. do Prado Leite, *On Non-Functional Requirements in Software Engineering*. Springer Berlin Heidelberg, 2009, pp. 363–379.

[21] M. Hassenzahl, "The effect of perceived hedonic quality on product appealingness," *International Journal of Human-Computer Interaction*, vol. 13, no. 4, pp. 481–499, 2001.

[22] G. C. Roman, "A taxonomy of current issues in requirements engineering," *Computer*, vol. 18, no. 4, pp. 14–23, 1985.

[23] I. Sommerville, *Software Engineering, 10th Edition*. Pearson, 2015.

[24] *ISO/IEC 25010: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*, ISO Std., 2011. [Online]. Available: https://www.iso.org/standard/35733.html

[25] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Springer US, 2000.

[26] J. Doerr, D. Kerkow, T. Koenig, T. Olsson, and T. Suzuki, "Non-Functional Requirements in industry - three case studies adopting an experience-based NFR method," in *Proc. of the 13th IEEE International Conference on Requirements Engineering*, 2005, pp. 373–382.

[27] M. Daneva, L. Buglione, and A. Herrmann, "Software Architects' Experiences of Quality Requirements: What We Know and What We Do Not Know?" in *Proc. of the 19th International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer Berlin Heidelberg, 2013, pp. 1–17.

[28] D. Ameller, C. Ayala, J. Cabot, and X. Franch, "How do software architects consider Non-Functional Requirements: An exploratory study," in *Proc. of the 20th IEEE International Requirements Engineering Conference*, 2012, pp. 41–50.

[29] D. Ameller, X. Burgues, D. Costal, C. Farre, and X. Franch, "Non-functional requirements in model-driven development of service-oriented architectures," *Science of Computer Programming*, vol. 168, pp. 18–37, 2018.

[30] *Proc. of the 2nd International Workshop on Software and Performance*. ACM, 2000.

[31] V. Cortellessa, H. Singh, and B. Cukic, "Early Reliability Assessment of UML Based Software Models," in *Proc. of the 3rd International Workshop on Software and Performance*. ACM, 2002, pp. 302–309.

[32] L. Grunske, B. Kaiser, and Y. Papadopoulos, "Model-Driven Safety Evaluation with State-Event-Based Component Failure Annotations," in *Proc. of the 8th International Symposium on Component-Based Software Engineering*. Springer Berlin Heidelberg, 2005, pp. 33–48.

[33] V. Cortellessa, A. Di Marco, and P. Inverardi, "Integrating Performance and Reliability Analysis in a Non-Functional MDA Framework," in *Proc. of the 10th International Conference on Fundamental Approaches to Software Engineering*. Springer Berlin Heidelberg, 2007, pp. 57–71.

[34] S. Wenzel, D. Poggenpohl, J. Jürjens, and M. Ochoa, "Specifying model changes with UMLchange to support security verification of potential evolution," *Computer Standards & Interfaces*, vol. 36, no. 4, pp. 776–791, 2014.

[35] N. Mani, D. C. Petriu, and M. Woodside, "Studying the Impact of Design Patterns on the Performance Analysis of Service Oriented Architecture," in *Proc. of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, 2011, pp. 12–19.

[36] M. Becker, M. Luckey, and S. Becker, "Model-driven Performance Engineering of Self-adaptive Systems: A Survey," in *Proc. of the 8th International ACM SIGSOFT Conference on Quality of Software Architectures*. ACM, 2012, pp. 117–122.

[37] J. Criado, S. Martínez-Fernández, D. Ameller, L. Iribarne, N. Padilla, and A. Jedlitschka, "Quality-Aware Architectural Model Transformations in Adaptive Mashups User Interfaces," *Fundamenta Informaticae Journal*, 2018, accepted for publication.

[38] J. Hutchinson, J. Whittle, and M. Rouncefield, "Model-Driven Rngineering practices in industry: Social, organizational and managerial factors that lead to success or failure," *Science of Computer Programming*, vol. 89, pp. 144–161, 2014.

[39] J. Hutchinson, J. Whittle, M. Rouncefield, and S. Kristoffersen, "Empirical Assessment of MDE in Industry," in *Proc. of the 33rd International Conference on Software Engineering*. ACM, 2011, pp. 471–480.

[40] R. Berntsson Svensson, T. Gorschek, and B. Regnell, "Quality Requirements in Practice: An Interview Study in Requirements Engineering for Embedded Systems," in *Proc. of the 15th International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer Berlin Heidelberg, 2009, pp. 218–232.

[41] R. Acerbis, A. Bongio, M. Brambilla, M. Tisi, S. Ceri, and E. Tosetti, "Developing eBusiness Solutions with a Model Driven Approach: The Case of Acer EMEA," in *Proc. of the 7th International Conference on Web Engineering*. Springer Berlin Heidelberg, 2007, pp. 539–544.

[42] M. Afonso, R. Vogel, and J. Teixeira, "From code centric to model centric software engineering: practical case study of MDD infusion in a systems integration company," in *Proc. of the 4th Workshop on Model-Based Development of Computer-Based Systems and 3rd International Workshop on Model-Based Methodologies for Pervasive and Embedded Software*, 2006, pp. 125–134.

[43] J. Whittle, J. Hutchinson, M. Rouncefield, H. Burden, and R. Heldal, "A taxonomy of tool-related issues affecting the adoption of Model-Driven Engineering," *Software & Systems Modeling*, vol. 16, no. 2, pp. 313–331, 2017.

[44] A. Cicchetti, F. Ciccozzi, and J. Carlson, "Software Evolution Management: Industrial Practices," in *Proc. of the 10th Workshop on Models and Evolution*. CEUR-WS, 2016, pp. 8–13.

[45] M. Daneva, A. Herrmann, and L. Buglione, "Understanding Quality Requirements Engineering in Contract-Based Projects from the Perspective of Software Architects: An Exploratory Study," in *Relating System Quality and Software Architecture*, I. Mistrik, R. Bahsoon, P. Eeles, R. Roshandel, and M. Stal, Eds. Morgan Kaufmann, 2014, pp. 325–357.

[46] A. Caracciolo, M. F. Lungu, and O. Nierstrasz, "How Do Software Architects Specify and Validate Quality Requirements?" in *Proc. of the 8th European Conference on Software Architecture*. Springer, 2014, pp. 374–389.

[47] M. Torchiano, F. Tomassetti, F. Ricca, A. Tiso, and G. Reggio, "Relevance, benefits, and problems of software modelling and model-driven techniques – A survey in the Italian industry," *Journal of Systems and Software*, vol. 86, no. 8, pp. 2110–2126, 2013.

[48] L. T. W. Agner, I. W. Soares, P. C. Stadzisz, and J. M. Simão, "A Brazilian survey on UML and model-driven practices for embedded software development," *Journal of Systems and Software*, vol. 86, no. 4, pp. 997–1005, 2013.

[49] G. Liebel, N. Marko, M. Tichy, A. Leitner, and J. Hansson, "Model-Based Engineering in the embedded systems domain: an industrial survey on the state-of-practice," *Software & Systems Modeling*, 2016.

[50] D. Ameller, M. Galster, P. Avgeriou, and X. Franch, "A survey on quality attributes in Service-Based Systems," *Software Quality Journal*, vol. 24, no. 2, pp. 271–299, 2016.

[51] P. Mohagheghi, W. Gilani, A. Stefanescu, M. A. Fernandez, B. Nordmoen, and M. Fritzsche, "Where does Model-Driven Engineering help? Experiences from three industrial cases," *Software & Systems Modeling*, vol. 12, no. 3, pp. 619–639, 2013.

[52] V. Kulkarni and S. Reddy, "Introducing MDA in a large IT con-

sultancy organization," in *Proc. of the 13th Asia Pacific Software Engineering Conference*, 2006, pp. 419–426.

[53] P. Baker, S. Loh, and F. Weil, "Model-Driven Engineering in a Large Industrial Context – Motorola Case Study," in *Proc. of the 8th Conference on Model-Driven Engineering Languages and Systems*. Springer Berlin Heidelberg, 2005, pp. 476–491.

[54] A. Borg, A. Yong, P. Carlshamre, and K. Sandahl, "The Bad Conscience of Requirements Engineering: An Investigation in Real-World Treatment of Non-Functional Requirements," in *Proc. of the 3rd Conference on Software Engineering Research and Practice in Sweden*, 2003, pp. 1–8. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-16932

[55] D. Ameller, X. Franch, C. Gómez, J. Araujo, R. B. Svensson, S. Biffl, J. Cabot, V. Cortellessa, M. Daneva, D. M. Fernández, A. Moreira, H. Muccini, A. Vallecillo, M. Wimmer, V. Amaral, H. Brunelière, L. Burgueño, M. G. ao, B. Schätz, and S. Teufl, "Handling Non-Functional Requirements in Model-Driven Development: An ongoing industrial survey," in *Proc. of the IEEE 23rd International Requirements Engineering Conference*, 2015, pp. 208–213.

[56] R. van Solingen, V. Basili, G. Caldiera, and H. D. Rombach, *Goal Question Metric (GQM) Approach*. John Wiley & Sons, Inc., 2002.

[57] D. A. Dillman, J. D. Smyth, and L. M. Christian, *Internet, Phone, Mail, and Mixed-Mode Surveys: The Tailored Design Method*, 4th ed. Wiley Publishing, 2014.

[58] P. Mayring, *Qualitative content analysis: theoretical foundation, basic procedures and software solution*. Social Science Open Access Repository (SSOAR), 2014. [Online]. Available: http://nbn-resolving.de/urn:nbn:de:0168-ssoar-395173

[59] J. Corbin and A. Strauss, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, 3rd ed. SAGE Publications, Inc., 2008.

[60] L. Chung and B. A. Nixon, "Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach," in *1995 17th International Conference on Software Engineering*, 1995, pp. 25–25.

[61] L. Guzmán, M. Oriol, P. Rodríguez, X. Franch, A. Jedlitschka, and M. Oivo, "How Can Quality Awareness Support Rapid Software Development? – A Research Preview," in *Proc. of the 23rd International Working Conference on Requirements Engineering: Foundation for Software Quality*, 2017, pp. 167–173.

[62] X. Franch, C. Gómez, A. Jedlitschka, L. López, S. Martínez-Fernández, M. Oriol, and J. Partanen, "Data-Driven Elicitation, Assessment and Documentation of Quality Requirements in Agile Software Development," in *Proc. of the 30th International Conference on Advanced Information Systems Engineering*, 2018, pp. 587–602.

[63] R. C. Martin, *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR, 2003.

[64] W. Behutiye, P. Karhapää, D. Costal, M. Oivo, and X. Franch, "Non-functional Requirements Documentation in Agile Software Development: Challenges and Solution Proposal," in *Procs. of the 18th International Conference on Product-Focused Software Process Improvement - PROFES*, 2017, pp. 515–522.

[65] F. Büttner, U. Bartels, L. Hamann, O. Hofrichter, M. Kuhlmann, M. Gogolla, L. Rabe, F. Steimke, Y. Rabenstein, and A. Stosiek, "Model-driven standardization of public authority data interchange," *Science of Computer Programming*, vol. 89, pp. 162–175, 2014.

[66] M. Ali Babar, L. Bass, and I. Gorton, "Factors Influencing Industrial Practices of Software Architecture Evaluation: An Empirical Investigation," in *Proc. of the 3rd International Conference on Quality of Software Architectures*, 2007, pp. 90–107.

[67] S. Becker, H. Koziolek, and R. Reussner, "The Palladio Component Model for Model-driven Performance Prediction," *Journal of Systems and Software*, vol. 82, no. 1, pp. 3–22, 2009.

[68] M. Nambiar, A. Kattepur, G. Bhaskaran, R. Singhal, and S. Duttagupta, "Model Driven Software Performance Engineering: Current Challenges and Way Ahead," *SIGMETRICS Performance Evaluation Review*, vol. 43, no. 4, pp. 53–62, 2016.

[69] J. Eckhardt, A. Vogelsang, and D. Méndez Fernández, "On the Distinction of Functional and Quality Requirements in Practice," in *Proc. of the 17th International Conference on Product-Focused Software Process Improvement*. Springer, 2016, pp. 31–47.

[70] B. Selic, "What will it take? A view on adoption of model-based methods in practice," *Software & Systems Modeling*, vol. 11, no. 4, pp. 513–526, 2012.

[71] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer, 2012.

[72] J. Whittle and J. Hutchinson, "Mismatches between industry practice and teaching of model-driven software development," in *Proceedings of the 2011th International Conference on Models in Software Engineering*, ser. MODELS'11. Springer-Verlag, 2012, pp. 40–47.