The Journal of Systems and Software 95 (2014) 176–193

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

On the use of software design models in software development practice: An empirical investigation

Tony Gorschek[a,*], Ewan Tempero[b], Lefteris Angelis[c]

[a] Blekinge Institute of Technology, Karlskrona, Sweden
[b] University of Auckland, Auckland, New Zealand
[c] Aristotle University of Thessaloniki, Thessaloniki, Greece

ARTICLE INFO

Article history:
Received 3 July 2013
Received in revised form 31 March 2014
Accepted 31 March 2014
Available online 12 April 2014

Keywords:
Software design models
Empirical industrial survey
Model-driven engineering (MDD, MDE, UML)

ABSTRACT

Research into software design models in general, and into the UML in particular, focuses on answering the question *how* design models are used, completely ignoring the question *if* they are used. There is an assumption in the literature that the UML is the de facto standard, and that use of design models has had a profound and substantial effect on how software is designed by virtue of models giving the ability to do model-checking, code generation, or automated test generation. However for this assumption to be true, there has to be significant use of design models in practice by developers.

This paper presents the results of a survey summarizing the answers of 3785 developers answering the simple question on the extent to which design models are used before coding. We relate their use of models with (i) total years of programming experience, (ii) open or closed development, (iii) educational level, (iv) programming language used, and (v) development type.

The answer to our question was that design models are not used very extensively in industry, and where they are used, the use is informal and without tool support, and the notation is often not UML. The use of models decreased with an increase in experience and increased with higher level of qualification. Overall we found that models are used primarily as a communication and collaboration mechanism where there is a need to solve problems and/or get a joint understanding of the overall design in a group. We also conclude that models are seldom updated after initially created and are usually drawn on a whiteboard or on paper.

1. Introduction

The use of design models when designing software, in particular by using the UML, has held the promise of quality improvements, automated code generation, improved problem solving and design capabilities, and in general improved productivity in software development (Mohagheghi and Dehlen, 2008; Schmidt, 2006; Booch et al., 1999; Fowler and Scott, 2004).

Since the inception of the UML in 1997 (OMG[1]) the UML has evolved as the spearhead of the modeling effort and is reported to be the de facto standard for use in industry (see e.g. Anda et al., 2006; Budgen et al., 2011; Dobing and Parsons, 2006; Grossman et al., 2005; Kobryn, 2002; Mohagheghi and Dehlen, 2008). More recently, model-driven engineering (MDE) has grown as a development methodology that promises improved productivity, product quality, and shorter lead time by increasing the level of abstraction and reducing the gap between problem and solution through use of design models (see e.g. Mohagheghi and Dehlen, 2008; France and Rumpe, 2007). The UML has also had substantial effort and resources put into it to better support MDE (Sliwa, 2004).

Although much research effort has been applied to the field, many books written (especially on the UML), and conferences held (e.g. MoDELS[2]), on modeling software, most of this has been focused on refining the modeling concepts, or on conducting empirical studies limited to how modeling is used (e.g. see Budgen et al., 2011; Damm et al., 2000; Dobing and Parsons, 2006; Petre, 2009). Very few, if any, studies exist investigating *to what extent* design models are used by practitioners (Budgen et al., 2011; Dobing and Parsons, 2006). One of the few studies commenting on the extent of design model use was presented by Cherubini et al. (2007), who concluded

* Corresponding author. Tel.: +46 455 385817.
E-mail addresses: tony.gorschek@bth.se, tony.gorschek@gmail.com, tgo@bth.se (T. Gorschek), e.tempero@auckland.ac.nz (E. Tempero), lef@csd.auth.gr (L. Angelis).

[1] http://www.omg.org

[2] http://www.modelsconference.org

http://dx.doi.org/10.1016/j.jss.2014.03.082
0164-1212/

that the "use of formal graphical modeling languages, such as UML, was very low."

Others have noted the lack of evidence. Budgen et al. (2011), in their systematic literature review, stated that the use of the UML seems to be a "given", and that researchers in the field seem reluctant to ask questions contradicting this assumption. This lack is surprising. The two main questions that need investigation in relation to the development and release of any new idea, whether it be a method, tool, or language, are the idea's usability and usefulness (Gorschek et al., 2006; Riemenschneider et al., 2002). That is, can the idea be used and does it scale outside a laboratory environment, what is the cost/benefit of use, and does it fit its purpose.

Although some of these concepts have been investigated, the core question of to what extent design models are used has not been addressed to any significant degree. This is even more surprising given the fact that the UML (and other) modeling "standards" have been available to industry for over 15 years, and are a part of most software engineering curricula world-wide (IEEE/ACM, 2004, 2009). To address this lack, this paper presents the results of a large scale empirical survey aimed at simply investigating *if*, and *to what extent*, software developers use design models. The survey had 3785 respondents.

The rest of the paper is structured as follows. Section 2 discusses the background and related work. Section 3 presents research methodology, including research questions, and a discussion of threats to validity and their mitigation. We then present our results and their analysis in Section 4. Finally Section 5 gives our conclusions.

2. Background and related work

There is considerable interest in the use of modeling in software development, the study of which is represented by the field of model-driven engineering (MDE). MDE has been seen as a means to manage complexity (Schmidt, 2006), improve productivity, shorten development time, and improve software quality (Mohagheghi et al., 2012). However, there is little empirical evidence of its use or effectiveness in industry (Mohagheghi and Dehlen, 2008; Mohagheghi et al., 2012).

Modeling has a long history of use in software development. Exactly what might be considered the first use of modeling depends on the definition used, but Structured Design (Stevens et al., 1974) and Jackson Structured Programming (Jackson, 1975) are good candidates, indicating modeling has been a known technique since the 1970s. Modeling of object-oriented software is more recent, but a number of methods such as Object-Modeling Technique (OMT) (Rumbaugh et al., 1991), the Booch method (Booch, 1991), and the Object-Oriented Software Engineering (OOSE) method (Jacobson, 1992) are all now over 20 years old. It was the number of different methods and notations that existed in the early 90's that led to the demand for a common approach, which began as the Unified Method 0.8 in 1995 and was adopted by OMG as the UML 1.0 standard in 1997. Fowler provided the earliest standard text for the UML (Fowler, 1997) in 1997, with the reference text coming out two years later (Booch et al., 1999). The UML is regarded as the most common modeling notation, and much of the research on modeling involves using the UML.

There have been many claims regarding how much it is used. Kobryn reports "UML has been widely accepted throughout the software industry"(Kobryn, 2002); Dobing and Parsons say it "has become widely accepted as a modeling standard" (Dobing and Parsons, 2006); Anda et al. say there is "widespread use of UML in industry" (Anda et al., 2006); Grossman et al. report the UML as having "tremendous popularity" (Grossman et al., 2005), and Budgen et al. describe it as a "de facto standard" (Budgen et al., 2011). Despite these claims, none of these papers, or other papers making similar claims, provide or cite evidence to support them, so it is difficult to really know to what degree the UML is actually used.

The UML has had more empirical investigation than MDE. Budgen et al. describe a Systematic Literature Review on empirical studies of the UML (Budgen et al., 2011). Their motivation was the concern that the UML was found to be difficult to use and so were interested in determining to what extent the effectiveness of the UML has been studied empirically. They found 49 relevant papers published up to the end of 2008.

According to Budgen et al. the largest category of papers (14.5 papers of 49) looked at various aspects relating to comprehension of the UML diagrams. The next largest category was metrics (12), that is, taking measurements of the UML diagrams in different ways and using them for different tasks. Model quality (7.5) and Methods and tools (7) were the next biggest, followed by Maintenance (2), Adoption (2), and Usability (1). Crucially, none of these categories directly examine the degree to which the UML is actually used. They all, to one degree or other, operated within a context where the UML is assumed or encouraged.

Some of the papers identified by Budgen et al. give some insight into empirical studies of the UML use, and so we discuss those in more detail. Anda et al. describe a case study in a company to determine the benefits and difficulties when introducing the UML (Anda et al., 2006). They report improvements in traceability, communication, code design, and testing, and identified difficulties choosing the appropriate type of diagram, dealing with interfaces between models, and the level of detail in models. For this study, the company had chosen to adopt a method based on the UML, however level of success was not reported.

Grossman et al. carried out a study to examine the claims that the UML leads to greater performance and claims regarding difficulty of its use (Grossman et al., 2005). Their study was carried out via a web-hosted survey whose 131 participants were recruited from various sources involving people who have experience with the UML. For their research question of interest to us, whether individuals perceive the UML as beneficial, they conclude that there was a positive perception towards the UML. As their participants were recruited from those with experience using the UML, their results probably can not be generalized to the whole population of software developers.

Nugroho and Chaudron also carried out a survey looking at how the UML is used and what issues there are with its use (Nugroho and Chaudron, 2008). They had 80 participants. They did not explain how they recruited their participants, but the implication is that they also targeted developers who were using the UML. More recently Petre presented an interview study with 50 software engineers at 50 companies Petre (2013). Of these 35 used no UML, no organisation wholeheartedly used UML, and of the remainder, 11 companies used UML in an informal way for as long as it was considered useful, after which it was discarded. Petre reports the participants as recruited opportunistically via her personal network, and observed "the sample may be biased slightly toward informants who had something to say about UML".

Empirical studies on modeling since 2008 follow a similar pattern. A representative sample can be found in a recent workshop on empirical studies in software modeling (EESSMod2011, 2011). The purpose of the empirical studies reported was on the effectiveness of modeling, and involved those who performed modeling, rather than whether or not models were used. Other studies include those by Hutchinson et al. (2011a,b), whose focus was on adoption of MDE, to understand how it is being adopted and what the social, organizational, and technical issues affect its success or failure, to provide organizations with empirical data to help them decide whether or not to use MDE. Their target population was users of MDE. They note that the hard criteria for inclusion in their

study was "the company must have been using models as a primary development artifact". They surveyed about 250 practitioners and carried out in-depth interviews of 22 MDE professionals.

One study of direct interest to us is a survey of 155 Italian software professionals carried out to determine their opinions and experience in modeling during software development (Torchiano et al., 2011; Tomassetti et al., 2012). Unlike most other studies, they did not try to sample only developers who have used MDE. However they had only 155 respondents, all Italian software professionals. Of the 155 respondents, 105 (68%) reported using modeling in software development.

To understand the degree to which modeling is performed, it is useful to understand *why* developers use models. Again, there is little direct empirical evidence, but a study by Cherubini et al. on why developers draw diagrams provides useful insight (Cherubini et al., 2007). They found that developers use diagrams of different fidelity and for different purposes, particularly for modeling, and suggest that the role of diagrams in software development is in some ways different to other engineering disciplines. Of particular interest to us, they observed that the "use of formal graphical modeling languages, such as UML, was very low." Their survey only included developers from one company, but it does raise the question as to whether this observation is generally true.

In summary, despite many claims of widespread use of modeling in software development, there has been little investigation into what the actual usage is in the general population of developers of object-oriented software.

3. Research methodology

This section introduces the research questions, the study design and execution, and threats to validity.

3.1. Research questions

In order to investigate to what extent modeling was used as a tool for development in design and coding, the following main research question was formulated.

RQ1: To what extent do developers use design models as a guide for development activities?

Based on this general research question, two sub-questions were formulated addressing the qualification of use, namely:

RQ1.1: Do the characteristics of (i) total years of programming experience, (ii) open or closed development, (iii) educational level, (iv) programming language used, and (v) development type, influence the use of design models?

RQ1.2: What are the main motivations or explanations qualifying the extent of use (see RQ1)?

3.2. Study design, operation, and analysis

To investigate these and other questions, we carried out a large-scale survey (Gorschek et al., 2010). The survey was executed through the creation of an on-line questionnaire that was designed using a mix of closed and open ended questions (Robson, 2002). The first version of the questionnaire was piloted using research programmers for the Computer Science department at the University of Auckland, New Zealand. As the test group took the survey, we monitored time, logged questions, and caught misunderstandings due to question formulation. We then followed with a debrief session. Based on the pilot, the survey instrument was improved. A second pilot of the instrument was performed using research colleagues at Blekinge Institute of Technology where four researchers (engineering and science PhDs) gave additional feedback on the instrument.

The motivation for using an on-line questionnaire was to maximize coverage and participation. Surveys are an appropriate strategy for collecting empirical results from a large population, and given an adequate response rate, an understanding of the population can be achieved (Punter et al., 2003). The questionnaire was made accessible on-line at http://surveymonkey.com between March and June of 2009. Participants were recruited primarily through personal contacts and forums targeted at software developers, and encouraging those who participated to spread the word. We provided information about the goals of the survey on our website (http://sefolklore.com), and posted a video to YouTube. The idea behind our information campaign was to get a "snowball effect", where "word-of-keyboard" spread information regarding the survey on the Internet. The campaign was successful in that the survey was eventually mentioned on twitter by a high-profile user, leading to a large number of respondents. The respondents only got a link to the survey, no additional information was disseminated.

The theoretical population (Gliner and Morgan, 2000) for the study was any and all software developers with experience in either closed or open source development and with experience in using any object-oriented programming language. The actual population, or sampling frame, was of course limited by Internet access and our ability to reach the developers within the given timeframe (and their willingness to participate). From one aspect the sample can be described as convenience sampling (Robson, 2002) as we utilized primarily our own contacts initially, however the sample quickly spread beyond our sphere of influence and contacts, with several hundred respondents before the twitter post (approximately 10 days after the survey went live), and thousands after it. A total of 4823 respondents started the survey, and 3785 completed all the mandatory questions, a completion rate of 78.5%. Given that the survey was substantial (three full pages, demanding about 15–20 min, and judging by the free-text responses a large number may have spent even more time), this completion rate suggests that participants were engaged in the survey and took it seriously.

From a sampling perspective it should be observed that we did not target subjects that we knew used design models, rather we targeted as widely as possible those professionals who had very good reasons to use design models in their work developing systems using object-oriented languages. As mentioned previously, most surveys and empirical investigations on modeling have a skewed sample – which we avoided as we presented the survey as an investigation of the use of object-oriented concepts, and not use of design models per se. Then, in the end of the survey the modeling question (the main focus of this paper) was posed. It was our intention to screen the respondents so that the modeling question was only answered by serious professionals utilizing object oriented concepts, and getting the respondents into the mindframe of object oriented concepts and software development before answering the modeling question.

The survey was aimed at investigating the perception, understanding and use of object oriented concepts in general, and not limited to only the use of design models. The survey had four parts. Part 1 gathered demographic information. Part 2 mainly addressed the concept of encapsulation, Part 3 covered class size, and Part 4 covered class depth. The questions pertaining to use of models was posed in the last part of the survey, with the deliberate intention of "weeding out" all respondents who were not relevant in terms of experience in using and working with object oriented development. It should also be noted that during the pre-tests (pilots) performed, all respondents understood Question 22 on use of design models (the main question analyzed in this survey) in a homogeneous way, and in the way we intended. That is, are

design models used as a precursor, inspiration, or "blueprint" for writing code (and "writing code" the respondents understood as development activities associated with implementation).

The survey featured 22 questions, of which 7 were for demographic information and 14 were on various aspects of object-oriented design. The results for the object-oriented design questions are discussed elsewhere (Gorschek et al., 2010). The last question (see below) is the subject of this paper. The non-demographic questions were, with two exceptions, multiple-choice on a scale ranging from "I always..." to "I never...". The exceptions were only free-text. However most of the multiple-choice questions, including the question discussed here, also had a free-text option to provide participants with an opportunity to provide qualifications of their answers to the questions with the pre-defined alternatives. The full survey is available on our website http://sefolklore.com.

Questions Q1–Q7 (demographics), and Q22 were the main input used for the purposes of this paper and the study of use of design models. The main survey question in relation to modeling was:

Q22: When you write code, to what degree do you use design models (e.g. UML diagrams) to guide you?

- Never (0%).
- Rarely (<10%).
- Sometimes (<25%).
- Less than half the time (<50%).
- More than half the time (≥50%).
- Much of the time (>75%).
- Almost all of the time (>90%).
- All the time (100%).

As the context of the survey overall was on analysis, design, and realization of software, Q22 was interpreted as "do you model" and not purely as "using models to guide programming". This was concluded based on the 1700+free-text responses provided by the respondents in relation to answering this question, however, for the purposes of this survey we focus on the use of design models as this was the question posed in the survey. In addition, even if most free-text responses confirmed that most respondents thought of the UML in their responses, we do not explicitly view the UML as the only representative of design models. Quite the opposite, we are very liberal in the interpretation of what constitutes a "model", and allow any type or level of formalism to be counted as a model, actually loading the results in favor of models and in favor of using design models.

The free-text responses were categorized using an exploratory categorization and coding schema based on open coding (Strauss and Corbin, 1998). That is, categories were created as the free-text answers were read, and similar statements were put into the same category. Existing categories were reformulated in light of newly-encountered answers. For example, the category "Only for complex designs" was created when three or more respondents stated an explanation similar to this statement. In cases where fewer than three respondents had a similar statement, or when interpretation of the statement was not possible, the responses were put in the "no explanation" category (56.7% of the respondents who did put in a qualifying statement ended up in this category). Details in relation to the categories created can be seen in Section 4.4. Under each category created, we quote several respondent answers to provide examples of what we placed in each category. The process of categorization was that one researcher (main categorizer) did the base categorization (creation and allocation of free-text answers to categories), then a second researcher reviewed the categorization and categories allocation of free-text statements, commented on it, discussed interpretations, and then the main categorizer updated the coding accordingly until agreement was reached.

For analysis purposes we use descriptive statistics as well as hypothesis tests to gauge statistical significance. Our main tools were contingency tables, Pearson's chi-squared test, and the Kendall's tau-coefficient (Sheskin, 2011). However we also used Correspondence Analysis (CoAn) (Greenacre, 1984). Generally, the main goal of CoAn is to describe the relationships between two categorical variables in a contingency table. These relationships are described by projecting the values of the variables as points on a two-dimensional space, in such a way that the resulting plot simultaneously describes the relationships between the categories of each variable. For each variable, the distances between points in the plot reflect the relationships between the categories. Similar categories are plotted close to each other while distant points show dissimilarity. The computations of the coordinates in the two-dimensional axis system are based on the chi-square statistic as measure of distance. The dependence is basically tested and studied with contingency tables, chi-square and CoAn. The Kendall's tau coefficient, although statistically significant in certain cases, was found to be very low in some cases. However, we report it since it gives us an indication of direction, or monotonicity, between the variables being tested. It actually helps us to determine where the low and high values of the main variable (in our case the use of design models) are directed with respect to the ordering of the other (e.g. experience, qualification, etc.).

3.3. Validity evaluation

We consider the four perspectives of validity and threats as presented in Wohlin et al. (2000).

3.3.1. Construct validity

The construct validity is concerned with the relation between the theories behind the research and the observations. The variables in our research are measured through the survey, including closed as well as open-ended questions where the participants are asked to share their professional experiences as developers.

Mono-operation bias can be a threat as only one question was posed in relation to use of models, however, during the pre-tests all subjects understood the question in a homogeneous manner, and in the manner we intended them to. In addition we used the free-text answers to get an indication of question misunderstanding. It should be observed, that even if many of the free-text answers were not categorized into the categories found in Section 4.4 we could still use them to gauge misunderstandings.

To avoid evaluation apprehension, complete anonymity of the subjects was guaranteed. There is always a risk that the background of the subjects (e.g. experience) is a central influence, however, due to the large sample, as well as the spread of competence and level of experience we feel the risk is limited.

Similarly, the large sample should mitigate any threats potentially caused by software engineering respondents having different personalities which previously been found to correlate with their values and attitudes to tools and methods they use (Feldt et al., 2010).

Hypothesis guessing (the respondents try to guess what the researchers want) is also a potential threat. The introduction to the survey (video and web page) stressed the importance of honesty, however this threat cannot be completely dismissed.

3.3.2. Conclusion validity

Threats to conclusion validity are concerned with the possibility of incorrect conclusions about a relationship in observations that may arise from error sources such as, instrumental flaws, influence posed on the subjects, or selection. We can not exclude the

possibility that the instrumentation (survey questions, formulations, explanations, etc.) were misunderstood by the subjects, however, pre-tests (pilots) and reviews by colleagues, as well as using the free-text clarifications to gauge question understanding, hopefully alleviated the risks of this threat.

The fact that only one question was used, albeit with a free-text clarification option also attached, can be seen as a threat as it did not enable the survey to catch more details about the use of design models, and the behavior of the developer respondents. However, we have been careful not to overstate our intent. We were interested in if the respondents use design models. Then the level of use was analyzed using different characteristics of the respondents. The "why" part was also covered through analysis of the free-text option. Asking more and extended questions on modeling would have yielded more data, but the only mitigation strategy as one question was asked was to focus on not over analyzing the results or overreaching in our conclusions as the area needs to be explored further.

Regarding subject influence, there can be a chance that some subjects interacted (e.g. colleagues at a work place) and that this interaction influenced some of the subjects' answers. However, due to the completion rate, as well as substantial sample size, we feel that the overall influence of this is limited.

The sample selected for the study were developers, however, we feel that the group was fairly heterogeneous (experiences, education etc.). In a small sample this might influence the outcome, however, due to sample size the risk of differences between subjects unduly influencing the result is low.

3.3.3. Internal validity

Internal validity is related to issues that may affect the causal relationship between treatment and outcome. Threats to internal validity include instrumentation, maturation and selection threats.

In our study, the instrument was pretested, as mentioned above. Maturation pertains to, for example, learning effect or subject's responses being influenced by boredom. Each subject participated once, thus learning effect was small, and the questionnaire took about 20 min to complete. In addition, the high completion rate of respondents (a clear majority of the respondents who started the survey also finished it) indicates an interest in participating, indicating that the respondents took an interest in being thorough in their efforts to answer the questions.

The interest of the subject may influence the representativeness of the subjects. This is a hard threat to counter as willingness to participate and interest in the subject are associated. The large sample may alleviate this to a degree, however the threat can not be dismissed. Further, the selection of subject was performed by using a wide range of media and channels, far beyond the control or sphere of influence of the researchers.

3.3.4. External validity

External validity is concerned with the ability to generalize the findings beyond the actual study. The actual setting of the study was an environment known to the subjects (from home/office using the web), thus our control and influence of the context was minimal. In addition, the sample was very similar to the population, that is, developers with experience in object-oriented programming.

Sampling is also a potential issue in external validity, as how the respondents were recruited could influence the answers. Although we used convenience sampling initially to spread the survey, as mentioned previously (see Section 3.2), the survey spread through programing forums and by word-of-keyboard. It is true that developers outside this "network" were excluded form participating. However, the characteristics of our respondents (our sample) is transparent (as seen under Results), thus the reader can judge generalizability. One additional aspect worth mentioning is that we deliberately did not see "modelers" and "modeling communities" as our population, rather we included any developer using object-oriented concepts, which we believe to be a less skewed sampling strategy.

Further, the sample is a volunteer sample. It can be argued that many potential respondents out of our population of developers developing OO systems fall out of scope (not in our sample) as they choose not to volunteer. This is a threat that is very hard to alleviate. Even if we got formal mandate to send the survey out to a random and representative sample of all companies (to their developers) in the world, there is no way of knowing how this would have alleviated the potential issue, as people can still choose not to participate. Also, any survey "forcing" a properly selected sample to respond might alleviate sampling issues of this nature, but introduce any number of other validity treats resulting in the non-voluntary nature of the recruitment of respondents. However, it is important to realize that our sample is a voluntary sample of OO developers.

Another potential threat in relation to sampling is that we do not know what type of industry our sample represents, or whether a specific company is overrepresented. We did ask about development type, programming language and other such demographic information in the survey, and so can assess to some degree what areas of the industry are represented. Asking for more specific information would have made anonymity harder, as well as introduce other threats such as evaluation apprehension.

Table 1
Geographical distribution of respondents.

Continent	Number	Percent
Africa	1	0.03%
Asia	294	7.77%
Australasia	426	11.25%
Europe	1207	31.89%
North America	1724	45.55%
South America	77	2.03%
Unknown	56	1.48%

4. Results and analysis

This section presents results from the survey, and is organized according to the research questions in Section 3, however we will begin with a summary of the demographics of the participants. The complete survey is available at http://sefolklore.com.

4.1. Respondent demographics

A total of 4823 respondents from 84 different countries began the survey, with 3785 completing the compulsory questions of the survey. Table 1 shows the distribution of responses by continent, indicating most of the responses originating from North America, with Europe a close second. Just over half the respondents had some open-source development experience, with the half of those having done 1–3 years of open source development. Almost all (94.5%) had undertaken closed-source development, with approximately half having done so for 1–8 years and 10% having done more than 20 years (see Fig. 1).

A variety of languages have been used by participants (Fig. 2) with the most common being C# (56%), Java (49%), C++(45%), or Python (21%). Almost all participants (94%) used one of these languages. About half (47.4%) of the respondents had experience in bespoke software development (Fig. 3) and almost all (95.6%) claimed experience in programming, with at least half claiming experience also in requirements, design, testing, and architecture (Fig. 4).

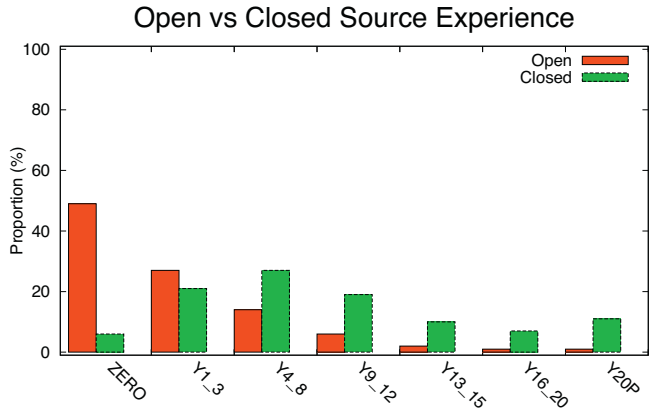

Fig. 1. Distribution of respondents based on years of open or closed source development experience.

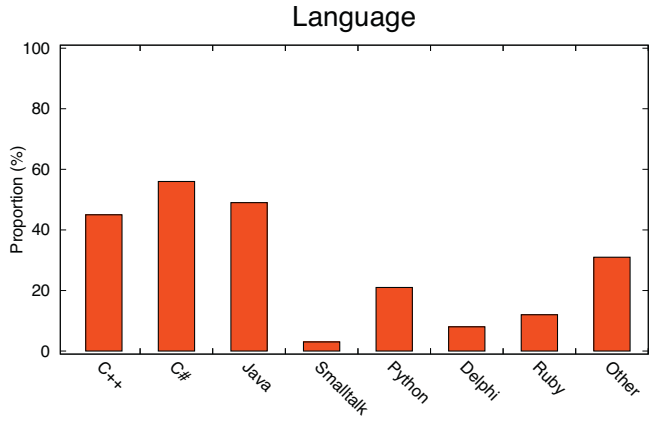

Fig. 2. Distribution of respondents based on language they feel comfortable doing development with. Respondents could choose multiple languages.

Regarding highest qualification (Fig. 5), half of the respondents had a Bachelor's degree (2420, 50%), 1245 (26%) had a Masters, 175 (5%) had a Ph.D., and about 10% had a trade certificates qualification or had taken professional development courses. About 9% (429) reported to have no formal training or qualification. For experience with operating systems (Fig. 6), most (3276 or 87%) were familiar with Windows, 2101 (56%) were familiar with Unix, and 789 (21%) and 140 (4%) were familiar with MacOS or other operating systems respectively.

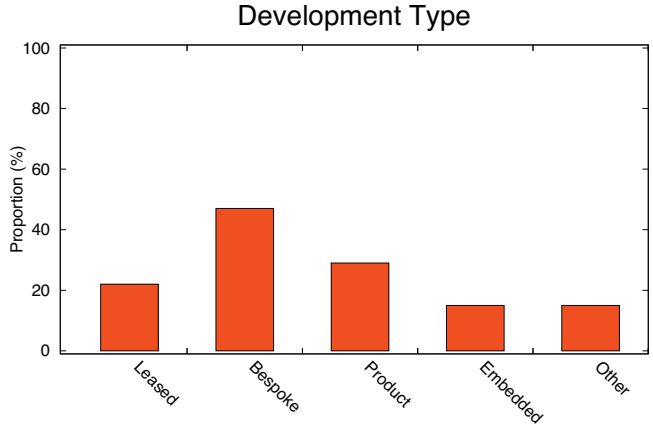

Fig. 3. Distribution of respondents development type they work with. Respondents could choose multiple types.

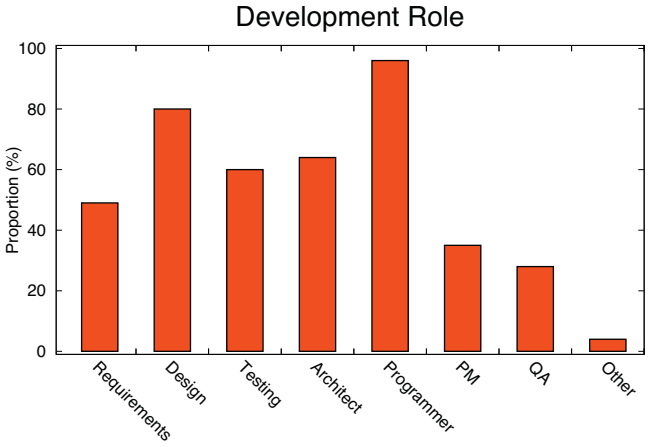

Fig. 4. Distribution of respondents base on development role. Respondents could choose multiple roles.

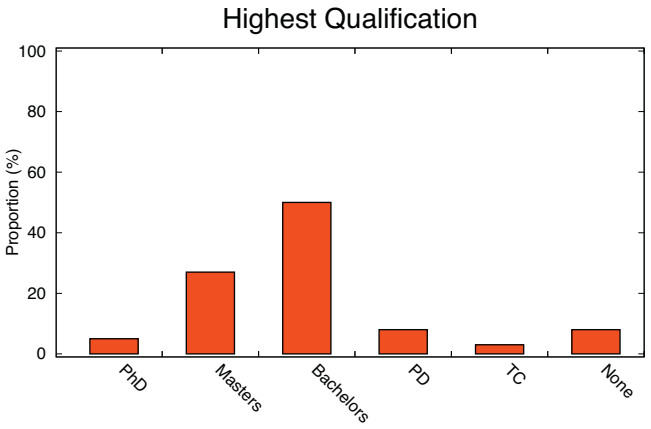

Fig. 5. Distribution of respondents based on highest qualification (PD is Professional Development courses, TC is Trade Certificate).

4.2. Extent of design model use (RQ1)

About half of the respondents (48.6%, see Fig. 7 and Table 2) never or rarely use design models as a guide for development. Almost 70% of the respondents use models in less than 25% of the cases, compared to about 11% that use models more than 75% of the time.

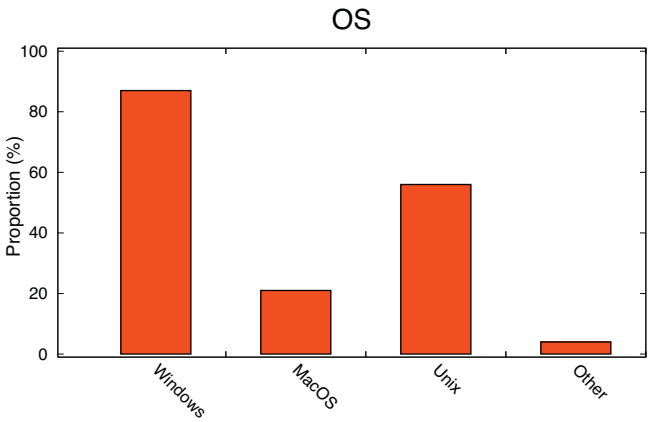

Fig. 6. Distribution of respondents based on operating system they feel comfortable developing in. Respondents could choose multiple operating systems.

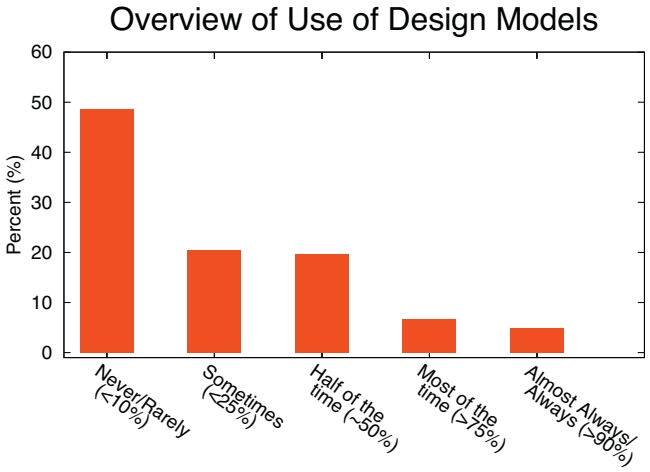

Fig. 7. Distribution of respondents according to their use of models.

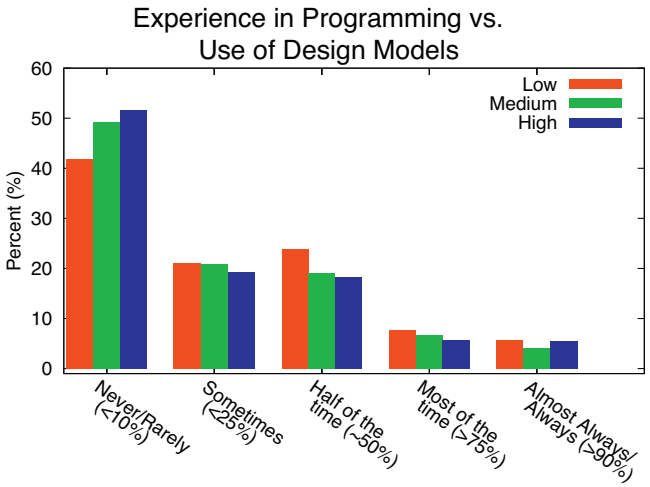

Fig. 8. Common distribution of "experience in programming" and "use of modeling"

The respondents' answers reflected a very liberal interpretation of "models" and "modeling", thus even less formal representations, and more ad-hoc modeling, was considered as doing modeling and using design models. For example, one respondent stated 'I have sketches', and another 'I rarely do this formally, but there is almost always a casual diagram on a whiteboard somewhere nearby'. Both of these were, for the purposes of our survey, counted as "using models". Thus if we limited the interpretation to a more formalized version of models (e.g. tool based UML notation) even fewer would have been classified as using design models.

As mentioned previously, very little work exists exploring the extent of design model use. One exception is a survey by Torchiano et al. reporting that about 68% out of 155 respondents did indeed use design models (Torchiano et al., 2011; Tomassetti et al., 2012). These results are quite different from ours. Possible explanations for this are their (in comparison) very limited sample, or that by their own admission their sampling method could have discouraged non-model users to answer their survey.

The implication of our results is that out of 3785 developers a clear majority rarely use design models, even informally. This does not only upset the assumption that the UML is the de facto standard, but also questions the use of modeling itself, independent of type of notation.

The next section investigates whether there is any connection between design model use and (i) total years of programming experience, (ii) open or closed development, (iii) educational level, and (iv) programming language used.

Table 2
Use of design models, overview.

	Frequency	Percent	Cumulative
Never/Rarely (<10%)	1838	48.6	48.6
Sometimes (<25%)	773	20.4	69.0
Half of the time (~50%)	745	19.7	88.7
Most of the time (>75%)	248	6.6	95.2
Almost Always/Always (>90%)	181	4.8	100.0
Total (valid answers)	3785	100.0	

4.3. Influencing factors – use of design models (RQ1.1)

4.3.1. Modeling and programming experience

The results for how use of models relates to programming experience is shown in Table 3 and Fig. 8, in which "Low" experience means less than 3 years in closed or open source programming, "Medium" more than 3 and less than 12 years, and "High" more than 12 years.

Looking at combined total programming experience of the respondents, experienced programmers seem to use models less than inexperienced programmers. This negative association between experience and use of models was confirmed as statistically significant (chi-square test $p = 0.001$, and Kendall's tau= -0.054 with $p < 0.001$).

Our decreasing trend contradicts the findings of Fitzgerald (1997), who identified a U-shaped curve indicating that developers with low and high experience use models more, while developers with medium level of experience use models less. Their thesis was that junior people need support in tools and models, but then get disenchanted as their experience grows, ending up using models again as experienced developers, but adapting them to their needs.

Our findings also contradict Davies et al. (2006) to some extent, as they identified rather an inverted U-shape, denoting medium experienced developers to be the most frequent model users.

Table 3
Contingency table for "experience in programming" and "use of design models".

Experience in Programming		Use of design models					Total
		Never/Rarely (<10%)	Sometimes (<25%)	Half of the time (~50%)	Most of the time (>75%)	Almost Always/Always (>90%)	
Low	Count	303	152	173	56	41	725
	%	41.8%	21.0%	23.9%	7.7%	5.7%	100.0%
Med.	Count	970	411	373	131	81	1966
	%	49.3%	20.9%	19.0%	6.7%	4.1%	100.0%
High	Count	565	210	199	61	59	1094
	%	51.6%	19.2%	18.2%	5.6%	5.4%	100.0%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100.0%

Table 4
Contingency table for "open source development experience" and "use of design models".

Open source (years)		Use of design models					Total
		Never/Rarely (<10%)	Sometimes (<25%)	Half of the time (~50%)	Most of the time (>75%)	Almost Always/Always (>90%)	
0	Count	925	370	360	135	82	1872
	%	49.4%	19.8%	19.2%	7.2%	4.4%	100%
1–3	Count	445	232	230	69	52	1028
	%	43.3%	22.6%	22.4%	6.7%	5.1%	100%
4–8	Count	255	107	98	26	32	518
	%	49.2%	20.7%	18.9%	5.0%	6.2%	100%
9–12	Count	128	49	39	13	6	235
	%	54.5%	20.9%	16.6%	5.5%	2.6%	100%
13–15	Count	44	8	12	2	4	70
	%	62.9%	11.4%	17.1%	2.9%	5.7%	100%
16–20	Count	24	3	5	1	3	36
	%	66.7%	8.3%	13.9%	2.8%	8.3%	100%
20+	Count	17	4	11	2	2	26
	%	65.4%	15.4%	3.8%	7.7%	7.7%	100%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100%

However, looking at high experience users our results are similar, that is, the more experience, the less use of models.

Overall, both the U-shape of Fitzgerald, and the inverted U-shape of Davis are contradicted by our results. Our results indicate a decreasing trend of modeling use as experience increases. Possible explanations for this range from one where Fitzgerald or Davis being correct, and our results being incorrect. However, our respondent base is substantially larger, and we have statistically significant results. In any case, the fact that out of a large sample experienced developers seem to use models less warrants further investigation. On a positive note it stands to reason that a good sample selection would be experienced developers as they could probably answer why they do or do not use models. Further, it is worth noting that programmers with low or medium level of experience, who are the ones using modeling the most, only do modeling about half of the time or less.

4.3.2. Models and open/closed source development experience

We wanted to investigate if experience in closed (CS) and open source (OS) environments influenced the use of design models, thus going beyond just years of experience in general. Looking at Table 4 we can see our respondents' experience in OS, cross tabulated with use of models. In Fig. 9 we can see their common distribution.

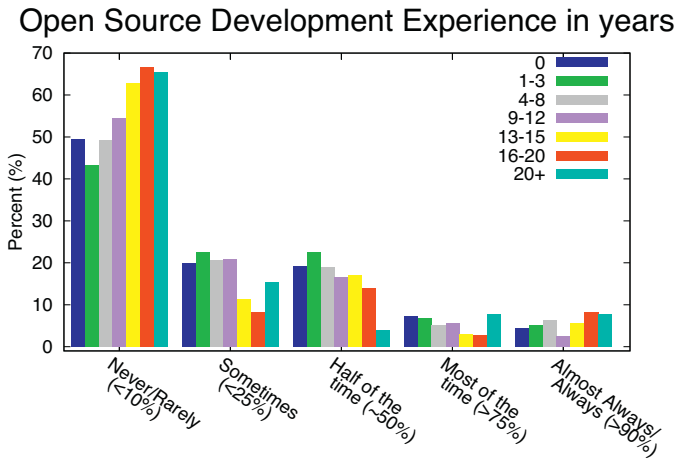

Fig. 9. Common distribution of "open source development experience" and "use of design models"

There is a statistically significant dependency between OS experience and the use of models ($p = 0.005$). However, the dependence is not directional (i.e. high OS experience does not imply high use of models, or vice versa). The lack of directional association is shown by the Kendall's tau-coefficient (tau = −0.013, $p = 0.364$), which is not statistically significant. The dependency actually means differences in the distribution of the use of models within the OS experience categories, and the distributions can be seen in Fig. 9. Noticeable is that the most experienced in OS (16–20 and 20+years) have a (relatively) high representation in the "Never/Rarely" group (66.7% and 65.4%), but at the same time more than well represented in the "Almost always/Always" group (8.3% and 7.7%). In addition, the most experienced OS developers (20+years) only have a 3.8% representation in the "Half of the time" group.

In order to investigate the association between OS experience and the use of models, we performed a Correspondence Analysis (CoAn) (Greenacre, 1984) on the contingency table (Table 4), as discussed in Section 3.2. Fig. 10 shows a high association between "1 and 3" years of OS experience and "Half of time" and "Sometimes" use models groups. We can also see that the three groups with the highest level of OS experience (13–15, 16–20, 20+years) are quite far apart from the each other and from the other four experience

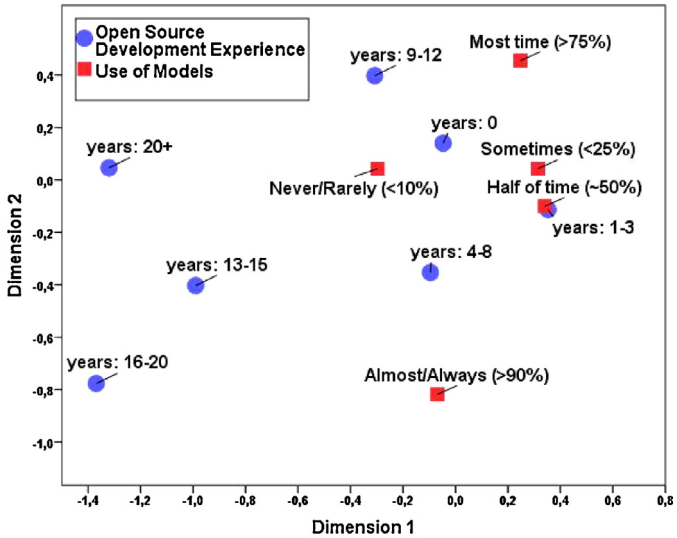

Fig. 10. CoAn Plot – "open source development experience" and "use of design models".

Table 5
Contingency table for "closed source development experience" and "use of design models".

Closed source (years)		Use of design models					Total
		Never/Rarely (<10%)	Sometimes (<25%)	Half of the time (~50%)	Most of the time (>75%)	Almost Always/Always (>90%)	
0	Count	104	36	44	13	12	209
	%	49.8%	17.2%	21.1%	6.2%	5.7%	100%
1–3	Count	341	159	179	57	43	779
	%	43.8%	20.4%	23.0%	7.3%	5.5%	100%
4–8	Count	493	209	207	82	38	1029
	%	47.9%	20.3%	20.1%	8.0%	3.7%	100%
9–12	Count	373	169	125	35	34	736
	%	50.7%	23.0%	17.0%	4.8%	4.6%	100%
13–15	Count	171	79	72	20	19	361
	%	47.4%	21.9%	19.9%	5.5%	5.3%	100%
16–20	Count	137	48	50	11	13	259
	%	52.9%	18.5%	19.3%	4.2%	5.0%	100%
20+	Count	219	73	68	30	22	412
	%	53.2%	17.7%	16.5%	7.3%	5.3%	100%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100%

groups, showing their differences concerning the distribution of use of models. However, the most interesting observation is that the most experienced groups are closest to both Never/Rarely and Almost/Always modeling groups.

The most experienced respondents seem to be polarized. They have chosen to either use models, or in principle to never use models. This seems to be confirmed by very few of the most experienced respondents being positioned in the middle of the spectrum (i.e. use models some or half of the time) as illustrated in the CoAn plot.

If we move to closed source (CS) development experience, seen in Table 5, a chi-square test shows that there is a significant dependence between CS experience and the use of models ($p = 0.040$). Also, there seems to be a low, but significant, negative dependency (Kendall's tau-coefficient, tau = −0.043 with $p = 0.002$). Looking at the clustered bar chart in Fig. 11 we can see that the highly experienced CS development groups (16–20 and 20+years) are rare users of models (52.9% and 53.2%), while frequent users are mostly found in the less experienced groups.

In general, when it comes to our respondents experienced in CS, the more experience they have, the less they tend to use models. This is not that surprising from one standpoint as experienced developers might not need models to guide them. However, looking at research into modeling, the assumption is not that only inexperienced programmers model and use models, rather that modeling can and should be used by all. Concepts such as MDE are not viable otherwise, as having models that are more or less complete and updated are a prerequisite. Further, the creation and (in multiple projects) reuse of models is paramount to attain the benefits with a reasonable (or at all positive) ROI on the modeling effort (Mohagheghi and Dehlen, 2008; Budgen et al., 2011). It should be observed that our question to the respondents was not "do you *need* modeling", rather we asked "do you use models" as our interest was in the state-of-practice and whether models are used. However, if we only focus on the "use" of design models this actually makes the response even more interesting, as independent of who creates the models, they are not used as a precursor for development (writing code/implementation).

4.3.3. Educational qualification and design model use

We thought it important to investigate the association between developers' level of formal qualification and their use of models, as one could argue that lack of education and training is a factor that hinders the use of models (Dobing and Parsons, 2006).

Fig. 12 gives an overview of the relationship between educational qualification and model use, and Table 6 shows the contingency table.

Regarding respondent qualification, there is a statistically significant positive dependency between use of models and qualification. Overall, the higher the qualification a developer has, the more likely the developer is to use models (chi-square test, $p < 0.001$, Kendall's

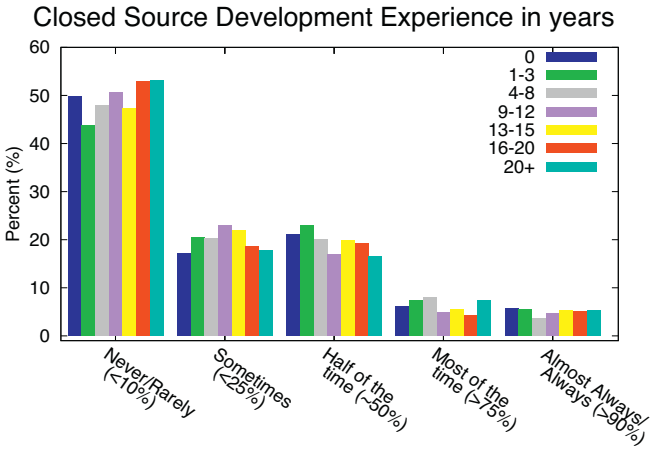

Fig. 11. Common distribution of "closed source development experience" and "Use of Design Models".

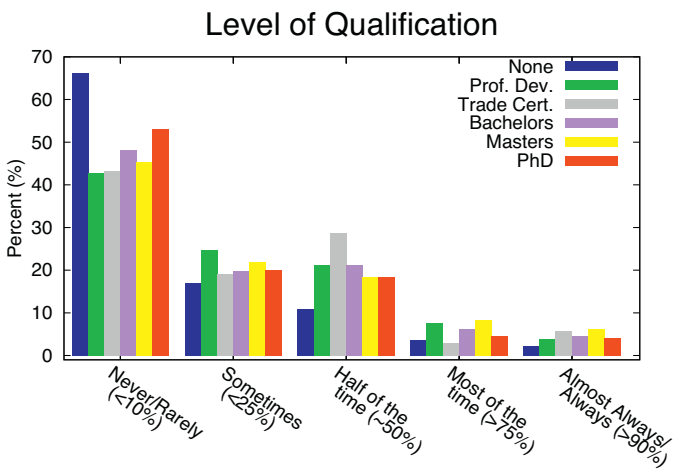

Fig. 12. Common distribution of "level of qualification" and "use of design models"

Table 6
Contingency table for "level of qualification" and "use of design models".

Highest qualification		Use of design models					Total
		Never/Rarely (<10%)	Sometimes (<25%)	Half of the time (~50%)	Most of the time (>75%)	Almost Always/Always (>90%)	
None	Count	200	51	33	11	7	302
	%	66.2%	16.9%	10.9%	3.6%	2.3%	100%
Prof.	Count	125	72	62	22	11	292
Dev.	%	42.8%	24.7%	21.2%	7.5%	3.8%	100%
Trade	Count	45	20	30	3	6	104
Cert.	%	43.3%	19.2%	28.8%	2.9%	5.8%	100%
Bachelors	Count	917	375	404	120	88	1904
	%	48.2%	19.7%	21.2%	6.3%	4.6%	100%
Masters	Count	458	220	184	84	62	1008
	%?	45.4%	21.8%	18.3%	8.3%	6.2%	100%
PhD	Count	93	35	32	8	7	175
	%	53.1%	20.0%	18.3%	4.6%	4.0%	100%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100%

tau-coefficient, tau= 0.043 with $p = 0.002$). This seems logical as most respondents with higher level of qualification should have been privy to courses and training in modeling and use of models, such as the UML, as most software engineering or equivalent curricula have some element of modeling.

Closer investigation of the level of qualification results shows that most of the respondents either had a Bachelor's or Master's degree (76%). However, the surprising thing is that most of these respondents ((917 + 375 + 458 + 320)/(1904 + 1008) = 71%) used models less than 25% of the time. Thus even if there is a connection between level of qualification and use of models, very few of the respondents, that we expect to have had the training, used models. One could argue that given the long history of modeling, including the UML (see Section 2), and that it has been a part of curriculum recommendations for some time (IEEE/ACM, 2004, 2009; Jensen et al., 1978), the lack of education in relation to modeling (Dobing and Parsons, 2006) ought not be the main reason for not using models for conceptual analysis and as a guide for development.

An overview can be seen in the CoAn plot (see Fig. 13) where Bachelors and Masters are closer to the "Sometimes" category. Also interesting to observe is the outliers of the respondents belonging to the "None" and the "Trade Cert." categories, which clearly lie away from the academically trained respondents. This suggests

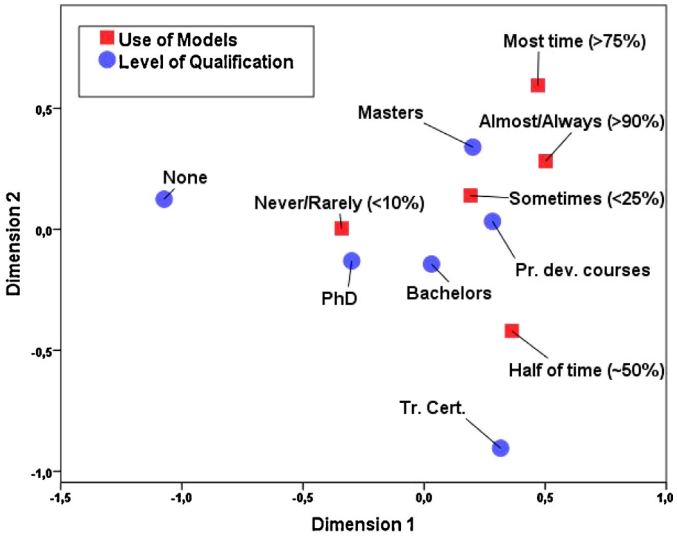

Fig. 13. CoAn – "level of qualification" and use of design models".

that academically trained professionals differ in modeling behavior in relation to the non-academically trained respondents. An interesting observation, although without statistical significance, is that respondents with a PhD as well as respondents with no education both lie close to the "never/rarely" category.

4.3.4. Programming language and use of design models

We investigated the respondent's programming language(s) experience, and the possible connection to the use of models. Specifically we asked respondents to identify which languages, from C++, C#, Java, Smalltalk, Python, Delphi, and Ruby, they are comfortable developing with. Table 7 shows the results.

Comparing the use of models in relation to the respondents' identified programming language we can see that for C++and C# developers there seems to be a positive dependency ($p = 0.001$), that is, respondents who mainly program in these languages have a statistically significant higher representation among active model users than respondents not using these languages. What is surprising is that for Java, Smalltalk and Delphi developers no such positive dependency could be established ($p > 0.4$). For Python and Ruby there seems to be an indication of the opposite, that is these groups have a higher representation in the "rare" (<10%) category (for Python $p = 0.001$, Ruby $p = 0.012$).

To the best of our knowledge, no other evidence has been presented that compares likelihood of, and attitude towards, modeling and programming language use. Based on our analysis programming language seems to be an influencing factor, with C++and C# programmers tending to support modeling, Java, Smalltalk, and Delphi programmers being neutral, Python and Ruby programmers tending not to support modeling. One possible explanation could be that developers working with C++and C# also generally work for organizations that have a more rigid tradition for documentation (and updating documentation), while developers working with Python and Ruby work for e.g. smaller companies that have no such traditions (or needs), e.g. start-ups. This theory would however not explain the apparent difference for Java, which should be as well established in larger organizations as C#. This phenomenon merits further study, especially for groups developing notation and tools.

4.3.5. Development type and use of design models

The last potential influencing factor investigated was development type. We asked the respondents which type of development activities they were most involved in, giving them following choices inspired by Lauesen (2002):

Table 7
Contingency table for "language" and "use of design models".

		Use of design models					
		Never/Rarely (<10%)	Sometimes (<25%)	Half of the time (~50%)	Most of the time (>75%)	Almost Always/Always (>90%)	Total
C++							
No	Count	1054	429	394	129	77	2083
	%	50.6%	20.6%	18.9%	6.2%	3.7%	100%
Yes	Count	784	344	351	119	104	1702
	%	46.1%	20.2%	20.6%	7.0%	6.1%	100%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100%
C#							
No	Count	900	316	295	90	74	1675
	%	53.7%	18.9%	17.6%	5.4%	4.4%	100%
Yes	Count	938	457	450	158	107	2110
	%	44.5%	21.7%	21.3%	7.5%	5.1%	100%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100%
Java							
No	Count	957	385	366	116	90	1914
	%	50.0%	20.1%	19.1%	6.1%	4.7%	100%
Yes	Count	881	388	379	132	91	1871
	%	47.1%	20.7%	20.3%	7.1%	4.9%	100%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100%
Smalltalk							
No	Count	1785	750	724	242	175	3676
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100%
Yes	Count	53	23	21	6	6	109
	%	48.6%	21.1%	19.3%	5.5%	5.5%	100%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100%
Python							
No	Count	1396	627	599	210	149	2981
	%	46.8%	21.0%	20.1%	7.0%	5.0%	100%
Yes	Count	442	146	146	38	32	804
	%	55.0%	18.2%	18.2%	4.7%	4.0%	100%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100%
Delphi							
No	Count	1701	710	685	223	162	3481
	%	48.9%	20.4%	19.7%	6.4%	4.7%	100%
Yes	Count	137	63	60	25	19	304
	%	45.1%	20.7%	19.7%	8.2%	6.3%	100%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100%
Ruby							
No	Count	1613	667	676	230	162	3348
	%	48.2%	19.9%	20.2%	6.9%	4.8%	100%
Yes	Count	225	106	69	18	19	437
	%	51.5%	24.3%	15.8%	4.1%	4.3%	100%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100%

1. Leased Consultancy (e.g. you sit at your company's customer site doing development for the customer)
2. Bespoke Development (Customer–Developer relationship, e.g. where you develop product that are ordered/commissioned by the customer but primarily sit at your company's site)
3. Off-the-shelf Products/Components Development (e.g. mass-market products, for example games, end-user software)
4. Embedded/Bundles Software Development (e.g. your software is a part of a larger product offering, for example a robot, a car, etc.).

The respondents could choose one or several options (as many companies provide multiple offerings).

The results are summarized in Table 8. Looking at the types of development respondents were involved in, we see, with two exceptions, very little indication that the type is a significant determinant as to the use of models. The respondents working with Leased development are relatively underrepresented in the rare category (chi-square test p=0.007). This might be explained with the fact that communication and coordination with the customer company in a leased scenario might imply legal and practical reasons for models being used as communication or contract elements. This ability to use models in communication was reported even pre-UML (Lubars et al., 1993; Luff et al., 1992), but there have also been confirmations that models using the UML enable communication and coordination (however mostly reported as beneficial for intra-development-team efforts) (Grossman et al., 2005).

The other exception was seen for the respondents working in an Off-the-shelf environment, which in contrast to the Leased case had a larger representation in the rare category (chi-square test p=0.05). This might be explained by time-to-market pressure of these types of development, and the impossibility to renegotiate deadlines as

Table 8
Contingency table for "type" and "use of design models".

		Use of design models					
		Never/Rarely (<10%)	Sometimes (<25%)	Half of the time (~50%)	Most of the time (>75%)	Almost Always/Always (>90%)	Total
Leased							
No	Count	1473	599	550	187	136	2945
	%	50.0%	20.3%	18.7%	6.3%	4.6%	100.0%
Yes	Count	365	174	195	61	45	840
	%	43.5%	20.7%	23.2%	7.3%	5.4%	100.0%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100.0%
Bespoke							
No	Count	1000	398	391	117	84	1990
	%	50.3%	20.0%	19.6%	5.9%	4.2%	100.0%
Yes	Count	838	375	354	131	97	1795
	%	46.7%	20.9%	19.7%	7.3%	5.4%	100.0%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100.0%
Off-the-shelf Product							
No	Count	1280	552	523	193	137	2685
	%	47.7%	20.6%	19.5%	7.2%	5.1%	100.0%
Yes	Count	558	221	222	55	44	1100
	%	50.7%	20.1%	20.2%	5.0%	4.0%	100.0%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100.0%
Embedded							
No	Count	1580	660	626	213	147	3226
	%	49.0%	20.5%	19.4%	6.6%	4.6%	100.0%
Yes	Count	258	113	119	35	34	559
	%	46.2%	20.2%	21.3%	6.3%	6.1%	100.0%
Total	Count	1838	773	745	248	181	3785
	%	48.6%	20.4%	19.7%	6.6%	4.8%	100.0%

in contractual scenarios (Gorschek et al., 2007; Aurum and Wohlin, 2005). Another explanation could be that the respondents understood the question as using COTS to develop their products (the question was posed as "What type of development are you currently working with"). If this is the case, use of models would not be that obvious as the components could be black-box, although interface models could be beneficial even in this case.

It is interesting to note that no statistically significant result was seen in relation to Bespoke development (p=0.075), which is the traditional contractually centered development form. One would expect that from a communication stand point (agreeing with the customer) models could be used, as in the case argued for Leased development. It could be that models are not used in Bespoke development because the potential for reuse (central for positive ROI Budgen et al., 2011; Mohagheghi and Dehlen, 2008) can be limited as many Bespoke projects are one-off development efforts (Aurum and Wohlin, 2005).

Respondents primarily associated with embedded development were not prominent in model use either (p=0.398). This is surprising as the complexity of embedded products seems to merit models being built and used (Edwards et al., 1997), particularly if one accepts the claims that models enables coordination activities. Also, embedded products would seem to be prime candidates for MDE (Mohagheghi and Dehlen, 2008; Grossman et al., 2005). However, we did not observe that respondents working with embedded software product development used models any more, or less, than the other groups.

4.4. Explanation and qualification – use of models (RQ1.2)

We have looked at to what extent models are used (RQ1), and investigated influencing factors of said use (RQ1.1). This section focuses on presenting and analyzing the last part of the survey, namely the respondents' own free-text qualifications in relation to how, why, or why not, and the extent of their model use. The answers were collected in free text form and analyzed through categorization (see Section 3.2 for details). A total of 1707 respondents filled in the question (about 45%), and out of these 845 answers were categorized into explanation categories (EC) as can be seen in Table 9. Answers such as "models are meaningless" and "models are great" were classified as "EC 0. No explanation" as we could not derive any relevant information based on them (or link them to an other EC). It should be observed that most respondents answered fairly shortly in the free-text option, and that most of the statements by the respondents that were carrying information qualifying their answers were indeed used by us in this analysis and categorized into one of the ECs.

Performing a chi-square test, we saw that the respondents in different groups of model use have a different distribution pertaining to the explanations they offer (p < 0.001). However, due to the larger part of the respondents being in the EC 0: "No explanation" we cannot state one single reason for models being or not being used. Extending the analysis to the use of CoAn, as seen in Fig. 14, we can however get some indications.

The respondents in the Almost/Always group (>90% use of modeling) seem closest to explanation EC 18: "UML enables the creation/help you write code". In the other extreme, the respondents in the Never/Rarely group (<10%) seem closest to EC 16: "Lack of good tool support". In addition, the respondents in the Sometimes group (<25%) seem closest to EC 2, 4, and 14.

We have chosen to focus the analysis around EC 1, 4, 6, 11, and 14 (denoted in gray rows in Table 9, all receiving 4% or more of the explanations offered by the respondents), however several EC's are covered indirectly in the analysis below.

4.4.1. EC 1. Only for very complex classes/designs, sometimes

Design models are powerful in terms of enabling the abstraction and clarification of the problem and potential solution (Selic,

Table 9
Contingency table for "explanation category" and "use of modeling".

Explanation category			Use of modeling					Total
			Never/Rarely (<10%)	Sometimes (<25%)	Half of the time (~50%)	Most of the time (>75%)	Almost Always/Always (>90%)	
0.	No explanation	#	455	167	138	54	48	862
		%	56.7%	49.4%	42.7%	44.6%	39.3%	50.5%
1.	Only for very complex designs, sometimes	#	25	33	24	7	4	93
		%	3.1%	9.8%	7.4%	5.8%	3.3%	5.4%
2.	Only relevant for larger development efforts	#	20	19	9	4	0	52
		%	2.5%	5.6%	2.8%	3.3%	0%	3.0%
3.	Natural language or comments is better	#	8	1	1	0	0	10
		%	1.0%	0.3%	0.3%	0%	0%	0.6%
4.	Only use initially then start coding (diagrams not kept/updated)	#	28	13	24	4	3	72
		%	3.5%	3.8%	7.4%	3.3%	2.5%	4.2%
5.	When I design something completely new	#	1	4	6	2	0	13
		%	0.1%	1.2%	1.9%	1.7%	0%	0.8%
6.	Enables visualization of the big picture/high level	#	13	14	20	19	12	78
		%	1.6%	4.1%	6.2%	15.7%	9.8%	4.6%
7.	Models help my thinking	#	7	6	23	6	8	50
		%	0.9%	1.8%	7.1%	5.0%	6.6%	2.9%
8.	Does not add any value	#	47	6	1	0	0	54
		%	5.9%	1.8%	.3%	0%	0%	3.2%
9.	Get outdated very early	#	16	6	0	0	0	22
		%	2.0%	1.8%	0%	0%	0%	1.3%
10.	Too much work/cost in relation to gain	#	25	4	2	0	0	31
		%	3.1%	1.2%	0.6%	0%	0%	1.8%
11.	Other type of models but not UML	#	42	15	22	9	9	97
		%	5.2%	4.4%	6.8%	7.4%	7.4%	5.7%
12.	Lack of training in modeling (or colleagues) can hinder communication	#	14	1	3	0	0	18
		%	1.7%	0.3%	0.9%	0%	0%	1.1%
13.	Use during refactoring mostly	#	3	0	2	0	0	5
		%	0.4%	0%	0.6%	0%	0%	0.3%
14.	Use models to communicate and coordinate with other developers	#	37	26	20	7	5	95
		%	4.6%	7.7%	6.2%	5.8%	4.1%	5.6%
15.	Model post coding of my or other peoples code for documentation purposes	#	16	5	6	1	5	33
		%	2.0%	1.5%	1.9%	0.8%	4.1%	1.9%
16.	Lack of good tool support	#	13	5	2	0	1	21
		%	1.6%	1.5%	0.6%	0%	0.8%	1.2%
17.	Not allotted time/customer not willing to pay for creation and continuous updates of models	#	22	8	9	1	1	41
		%	2.7%	2.4%	2.8%	0.8%	0.8%	2.4%
18.	UML enables the creation/ help you write code	#	2	1	8	7	26	44
		%	0.2%	0.3%	2.5%	5.8%	21.3%	2.6%
19.	We should model more	#	9	4	3	0	0	16
		%	1.1%	1.2%	0.9%	0%	0%	.9%
	Total	#	803	338	323	121	122	1707
		%	100%	100%	100%	100%	100%	100%

1998). One respondent stated 'I use them when they are available or already done, mostly, I won't do them except when dealing with something complex/architecture-related', another stated 'If it's a particularly complicated problem I'll sometimes do a class or activity diagram on a piece of scrap paper'. This seems to indicate that the developers use models (mostly referring to the UML explicitly), for conceptual analysis and problem solving, or as one respondent indicated 'I find them very useful to visualize the problem'. Visualization of the problem and potential solution can also enables improved problem solving efficiency (Harel, 1992; Schauer and Keller, 1998).

This also relates to EC7 where one respondent stated 'Models help explore concepts properly', indicating that several concepts and solutions could be compared as a step in the design. However it is interesting to note, very few used any sort of tool, or saved the models, rather they are throwaway work products. This relates to EC 4 below, and the ones that indicated not only model use but also creating models.

4.4.2. EC 4. Only use modeling initially then start coding (diagrams not kept/updated)

One respondent stated 'Only in the beginning of design. Later on, no more design models are used. . .', another stated 'it is good for initial very high level stuff'. Most respondents in this EC focused on creating and using models initially, but did not update the design models after coding commenced as expressed by one respondent

Table 10
Contingency table for "notation" and "use of design models".

Notation			Use of design models					Total
			Never/Rarely (<10%)	Sometimes (<25%)	Half of the time (~50%)	Most of the time (>75%)	Almost Always/Always (>90%)	
0	No information	#	565	240	223	86	78	1192
		%	77.2%	78.2%	75.6%	74.8%	67.8%	76.2%
1,2	UML/UML like	#	31	22	36	11	19	119
		%	4.2%	7.2%	12.2%	9.6%	16.5%	7.6%
3	Personal notation, less formal than e.g. UML	#	78	29	33	15	15	170
		%	10.7%	9.4%	11.2%	13.0%	13.0%	10.9%
4	NL	#	16	2	2	1	2	23
		%	2.2%	0.7%	0.7%	0.9%	1.7%	1.5%
5	Refactoring of code, not upfront design with models	#	11	1	0	0	0	12
		%	1.5%	0.3%	0%	0%	0%	0.8%
6	TDD	#	23	9	1	2	1	36
		%	3.1%	2.9%	0.3%	1.7%	0.9%	2.3%
7	Prototypes/mock-ups	#	8	4	0	0	0	12
		%	1.1%	1.3%	0%	0%	0%	0.8%
	Total	#	732	307	295	115	115	1564
		%	100%	100%	100%	100%	100%	100%

stating 'I use UML to visualize and design the class structure, but once coding starts to gain momentum, the UML is left behind'. It seems that the respondents in this group focused on modeling as a visualization enabler initially, but did not go for model transformation, even in the simplest forms, such as creating stubs. France and Rumpe (2007), in their paper about all types of models, presented several challenges in their roadmap for MDA/MDE, where issues related to inadequate tool support, but also inherent problems with transformation were discussed. Without the visualization and intuitive support of a tool, surpassing the flexibility of pen and paper (used by many respondents), there needs to be powerful code generation capabilities. Even if these do exist in some application areas (Glück and Visser, 2011), the potential loss of the free form visualization environment of something like a whiteboard seems to weigh more heavily (this is also confirmed below in EC 14).

4.4.3. EC 6. Models enable visualization of the big picture/high level

At a first glance this EC seems closely related to EC 1 and 4 as it is about visualization, and getting an overview of the problem and solution. However looking at some examples of respondents' statements we observe that it is specifically in relation to visualizing and getting an overview of dependencies and relationships between entities. For example, one respondent stated 'Usually only for the overall ("big picture") design to help create a mental picture of what is required and how it needs to interact', another stated 'I can picture the relationships between up to 3 classes at once. Beyond that, it's hard to track without a diagram of some sort'. That is, larger systems with many dependencies seemed to be candidates for an overview model to enable developers to identify dependencies.

4.4.4. EC 11. Other type of design/models but not UML

A substantial proportion of the respondents mention the UML by name in their natural language clarifications, but many to say that the type of notation used is "UML like", i.e. not the UML per se. One respondent stated 'Easier to rub something out with a pencil than re-code it. No UML though'. Looking at Table 10 and Fig. 15 we can see that about 7.6% of the respondents who clarified their use of notation specifically stated using the UML or UML like notation, while almost 11% stated using their own less formal notation, illustrated by one respondent stating 'I'll do designs, but I can't remember ever doing a UML diagram outside school'.

Performing a chi-square test we get statistically significant results ($p < 0.001$). Frequent modelers (>90%) tend to use the UML/UML like notations (as can be seen in Fig. 15), while the span between rare (<10%) to Most of the time (>75%) are closer to less

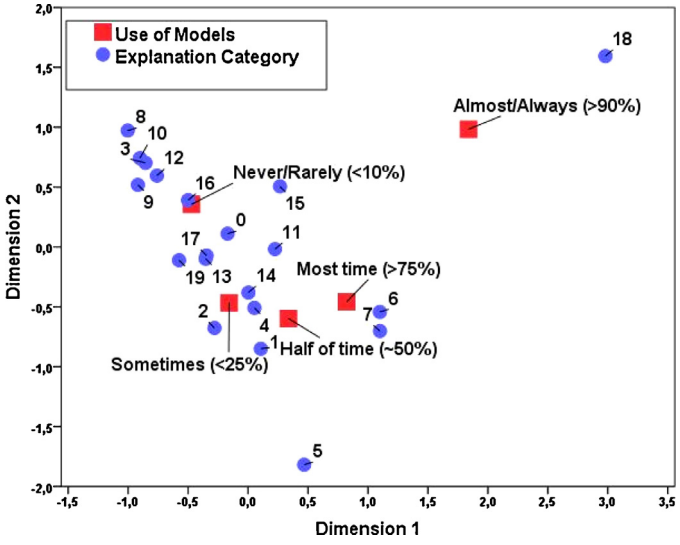

Fig. 14. CoAn – "explanation category" and "use of design models".

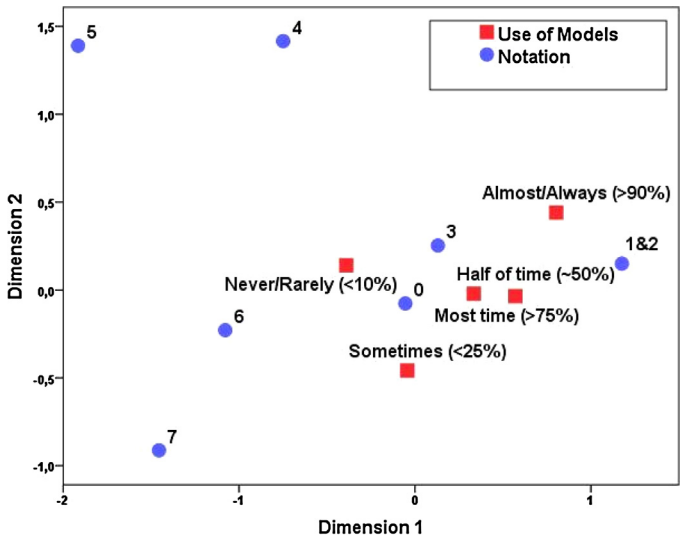

Fig. 15. CoAn – "Notation" and "Use of Design Models".

Table 11
Contingency table for "medium" and "use of modeling".

Medium		Use of design models					Total
		Never/Rarely (<10%)	Sometimes (<25%)	Half of the time (~50%)	Most of the time (>75%)	Almost Always/Always (>90%)	
No information	#	557	239	233	88	101	1218
	%	86.2%	84.8%	85.0%	80.7%	87.8%	85.4%
Tool	#	1	1	2	4	5	13
	%	0.2%	0.4%	0.7%	3.7%	4.3%	9%
Paper/Whiteboard	#	75	37	33	16	8	169
	%	11.6%	13.1%	12.0%	14.7%	7.0%	11.9%
In my head	#	13	5	6	1	1	26
	%	2.0%	1.8%	2.2%	0.9%	0.9%	1.8%
Total	#	646	282	274	109	115	1426
	%	100%	100%	100%	100%	100%	100%

formal notations. Looking specifically at the UML (and not modeling in general) our results seem to indicate that the UML is not the de-facto standard used, rather engineers use a mix of (informal) personal notations instead.

Several respondents mention the use of CRC cards (Class Responsibility Collaboration, Beck and Cunningham, 1989), ERDs (Entity-relationship Diagrams, (Chen and Mar, 1976)), and Data flow diagrams (Stevens et al., 1974), and also often explicitly stated that they do not use formal notations in any way. It is important to remember that these respondents were spread over the spectrum of model users, that is their definition of models included using informal "boxes and arrows" as stated by one respondent. Many of the avid modelers (>50%) made statements like 'I don't use strict UML diagrams, but I do use similar diagrams most of the time'.

Several reports from industrial application note that the UML is too big and complex (Dobing and Parsons, 2006; Ambler, 2002), which could be one explanation of using informal variants of notation other than the UML for modeling purposes. Another possible explanation could be the qualification (knowledge) of the developers, that is, their lack of training (Dobing and Parsons, 2006). Grossman et al. reported on a survey conducted yielding 131 responses from developers using modeling (Grossman et al., 2005). Their results indicated that over 40% received formal training, but an even greater proportion were self-taught. Looking at our respondents (see also Section 4.3) over 71% had at least a Bachelors degree. This would indicate that even if the UML was not part of the curriculum, some sort of modeling should have been. Looking at EC12 in Table 9 only about 1% discussed lack of training to be a central issue hindering the use of models.

4.4.5. EC 14. Use models to communicate and coordinate with other developers

A total of 5.6% of the respondents indicated that modeling is used primarily as a tool to communicate and coordinate in groups of developers, as clarified by one respondent stating 'Design models are communication tools'. Several empirical studies related to experiences from modeling in industry (see e.g. Grossman et al., 2005; Anda et al., 2006) confirm models as communication enablers. Software design is inherently a collaborative effort where whiteboards or simple pieces of paper are used as a brainstorming tool (Damm et al., 2000; Craft and Cairns, 2006). The informal nature of the medium used is also central, as can be seen in Table 11 where the use of an actual tool seems negligible (less than 1% of our respondents indicated using any sort of tool), while almost 12% indicated using whiteboard/paper as a medium. The ones using the whiteboard least are the Almost/Always modelers (>90%) ($p = 0.001$).

The type of medium used for modeling was determined from the free text answers, as we did not explicitly ask for what medium used. Rather we extracted it from statements such as "Design models are good for whiteboards and communicating within the team'. This is why 85% were classified as "No information". However, it should be observed that many of our respondents indicated that design models were abandoned after initial brainstorming sketches. Looking at Table 9 and combining EC 3–4, 8–10, and 15–16 (ranging from models created post coding through e.g. Javadoc, to lack of good tool support) one could infer that tool use is relatively limited. If so the implications could be substantial as even informal and/or incomplete models cannot be used for recordkeeping or documentation (as indicated by some, see e.g. Petre, 2009; Forward and Lethbridge, 2008). The implications for actually using models in a larger more extensive context (e.g. MDE) reach even further as completeness, formalism and tool use are pre-requisites, and there is even call for increasing formalism to enable effective and efficient MDE (France and Rumpe, 2007).

The CoAn analysis in Fig. 16 further shows that many of the respondents answer that they keep the design (and do it) in their heads. One respondent stated that 'I do it all in my head, only hit up a white board if others are involved'. Although this specific respondent was in the Almost/Always (>90%) modeling group, most using this medium belong to the Never/Rarely (<10%) group.

Another respondent stated 'When I do OO development, UML diagrams are valuable for communication among developers (current and future) and stakeholders', indicating that using design models for communication might go beyond fellow developers. The usefulness of modeling artifacts (such as UML activity diagrams) was tested by Thomson et al. (2008), indicating that the cost of creating models could be spread over several central activities beyond design and documentation, i.e. elicitation and communication with e.g. customers. The spreading of modeling cost over several development phases can be an effective means to justify the cost of modeling (Fricker et al., 2010).

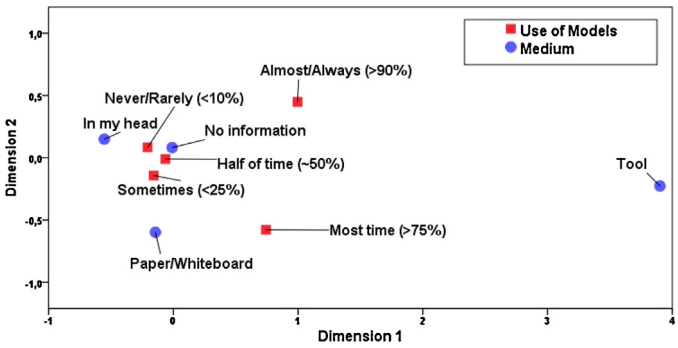

Fig. 16. CoAn – "Medium" and "Use of Modeling".

4.5. Validity threats revisited

While we discussed threats to validity and our mitigation of them in Section 3.3, it is worth revisiting them in light of our data and results. There are two main issues regarding the validity of our results. One is that there was only one question in the overall survey specific to RQ1. The other issue is to what extent our sample is representative of the developer population.

Asking a single question means we may not necessarily have all of the relevant context for the participants answers, and so there are limits to what extent we can generalize our results. Nevertheless, the answers to that question, together with the analysis of the free text commentary, show a clear trend, one that deserves further research.

Regarding how representative our sample was, as noted in Section 3.2 we had little influence on who participated. Also, it is difficult to construct a scenario where there was a bias against developers who used design models. The question we asked was the last question of the survey, and there was nothing in the material describing the survey indicating any question regarding modeling would be asked (see http://sefolklore.com for what participants saw). We see no reason why those who used modeling would be less inclined to participate than any other group of developers.

We also see nothing in the demographics to indicate that there was a significant bias in what kind of developer participated. We see a good spread of experience (Fig. 1), a variety of languages used (Fig. 2), most types of development (Fig. 3), with development done on all the standard operating systems (Fig. 6). If use of models tended to be role-specific, then we should have had a good sample (Fig. 4). If level of qualification was a factor, it did not show up in our analysis (e.g. Table 6), which was also true of the other demographic data.

5. Conclusions

While there have been other empirical studies on the use of design models (particularly of use of the UML), ours is distinguished in two main ways: our goal was to determine the extent to which design models (indirectly the UML) are used, rather than its effectiveness, and our sample was from a population of developers, rather than a population of modelers. In fact, there was no indication in the description of the survey that there would be any questions on modeling, and the modeling question was the last in the survey. As a consequence we believe that our sample is neither biased for, or against, the use of modeling, thus our sample is representative of the developer population, rather than the modeler population.

Our main conclusion is that, based on the 3785 developers that responded to the survey, about 50% *Never (<10%)*, about 70% *Rarely (<25%)* use design models – and in contrast only about 11% use models more than 75% of the time (see RQ1). Since both our and the respondents' definition of "what constituted a model" and what constitute "using models" was very liberal, this is a rather important result. The use of models in general, and the UML in particular, does not seem to be standard practice and de facto standard in software development – which challenges the assumption on which much of current research is based. The implications being far-reaching and substantial.

Model-driven engineering is based on the assumption that developers model, and see benefit in modeling and using models. We can not say with certainty that our respondents do not model, but we can say they very seldom use models. Our results show that most developers do not use models most of the time, and even fewer update their models beyond initial design sketches – which are done on the white board, and not in tools. Of course there are developers and companies that use models, and even model to its fullest extent by applying MDE. However, the widespread use of at least basic modeling is a pre-requisite for the spread of model-drive engineering. Based on our survey, this pre-requisite is not being met.

Studying the characteristics of our respondents (as per RQ1.1) we saw a number of surprising results. Experience level was negatively associated with degree of model use. This not only contradicts previous research, but also creates questions as to why we observed a decreasing trend. We did not find that senior developers rediscover models as they mature in experience, and used selected parts. Rather they use models less and less. Why this is exactly we cannot conclude based on the survey, but we can speculate that model use might not be seen as being beneficial by experienced developers, which is the worst possible interpretation as they should be in a good position to gauge likely benefits. Further investigation is needed to find a definite explanation.

In terms of education level and modeling we found what we expected. That is, developers with academic training model more than developers without it. This would seem to indicate that the curriculum in engineering education does have an impact. However, even here we see that about 71% of the developers in the "highly" educated group use models less than 25% of the time. This can be due to a lack of training, however this is not a likely conclusion as, based on the free-text responses, training was not considered central by many (see EC12, 1.1%, Table 10). We believe (albeit without evidence) that training can be a factor, but also in relation to being able to convey the potential benefits of using models to developers.

The connection between primary programming language used and the use of models is largely uninvestigated in research. We found that there seems to be a connection – where C++and C# developers use models more than others. While for Java there did not seem to be any dependence at all, which itself is interesting as one might expect Java to be in the same category as C#. The use of programming language and models is also identified through this study as an interesting avenue for future research.

Overall, based on the free-text responses, we found that models are used primarily as a communication and collaboration mechanism where there is a need to solve problems and/or get a joint understanding of the overall design in a group. We also conclude that models are seldom updated after making initial drawing(s), most often on a whiteboard or on paper.

The purpose of this study was not to critique or value the use of design models or the UML, but rather to investigate the assumption that models are used in industry. As the use seems to be limited, to say the least, and the level of use seems to be very informal, researchers in the field need to take a step back and contemplate why this is. More empirical studies in relation to usability and usefulness, as well as scalability, need to be performed – then the results, however disconcerting, need to be taken into account when developing both notations, deciding level of formalism, and designing tools. Continuing in the pursuit of exotic concepts and extending modeling standards such as the UML needs to be tempered by insights into cost/benefit and fitness for purpose, to get industry to buy in to the potential value (which we can not assume, but have to gather empirical evidence for). Then we need to investigate usability aspects to make sure that notation and tools are fit for purpose, to get the developers interested.

Acknowledgements

First and foremost we would like to thank the some 4823 developers who took time out of their busy lives to answer the survey. Thanks everyone – the frankness and honesty of the paper is for you! We researchers need to listen more and invent less. Second,

we would also like to thank Prof. Dr.Jürgen Börstler for his kind help — pointing us in the right direction of related work.

References

Ambler, S., 2002. Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process. John Wiley & Sons, Inc., New York, NY, USA.

Anda, B., Hansen, K., Gullesen, I., Thorsen, H., 2006. Experiences from introducing UML-based development in a large safety-critical project. Emp. Softw. Eng. 11, 555–581, http://dx.doi.org/10.1007/s10664-006-9020-6.

Aurum, A., Wohlin, C. (Eds.), 2005. Engineering and Managing Software Requirements. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Beck, K., Cunningham, W.,1989. A laboratory for teaching object oriented thinking. In: Conference Proceedings on OBJECT-oriented Programming Systems, Languages and Applications. OOPSLA'89. ACM, New York, NY, USA, pp. 1–6, http://dx.doi.org/10.1145/74877.74879.

Booch, G., 1991. Object-Oriented Design with Applications. Benjamin Cummings, Redwood City, CA, USA.

Booch, G., Rumbaugh, J., Jacobson, I., 1999. The Unified Modeling Language User Guide. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

Budgen, D., Burn, A.J., Brereton, O.P., Kitchenham, B.A., Pretorius, R., 2011. Empirical evidence about the UML: a systematic literature review. Softw. Pract. Exp. 41 (4), 363–392, http://dx.doi.org/10.1002/spe.1009.

Chen, P.P.-S., 1976. The entity-relationship model: toward a unified view of data. ACM Trans. Datab. Syst. 1 (March (1)), 9–36, http://dx.doi.org/10.1145/320434.320440.

Cherubini, M., Venolia, G., DeLine, R., Ko, A.J.,2007. Let's go to the whiteboard: how and why software developers use drawings. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI'07. ACM, New York, NY, USA, pp. 557–566, http://dx.doi.org/10.1145/1240624.1240714.

Craft, B., Cairns, P., 2006. Using sketching to aid the collaborative design of information visualisation software – a case study. In: IFIP International Federation for Information Processing, 2006, Vol. 221, Human Work Interaction Design: Designing for Human Work. Vol. 221, pp. 103–122.

Damm, C.H., Hansen, K.M., Thomsen, M.,2000. Tool support for cooperative object-oriented design: gesture based modelling on an electronic whiteboard. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI'00. ACM, New York, NY, USA, pp. 518–525, http://dx.doi.org/10.1145/332040.332488.

Davies, I., Green, P., Rosemann, M., Indulska, M., Gallo, S., Sep. 2006. How do practitioners use conceptual modeling in practice? Data Knowl. Eng. 58 (3), 358–380, http://dx.doi.org/10.1016/j.datak.2005.07.007.

Dobing, B., Parsons, J., 2006. How UML is used. Commun. ACM 49 (May (5)), 109–113, http://dx.doi.org/10.1145/1125944.1125949.

Edwards, S., Lavagno, L., Lee, E., Sangiovanni-Vincentelli, A., mar 1997. Design of embedded systems: formal models, validation, and synthesis. Proc. IEEE 85 (3), 366–390.

EESSMod2011, 2011. First International Workshop on Experiences and Empirical Studies in Software Modelling. http://www.eesmod.org

Feldt, R., Angelis, L., Torkar, R., Samuelsson, M., 2010. Links between the personalities, views and attitudes of software engineers. Inf. Softw. Technol. 52 (June (6)), 611–624, http://dx.doi.org/10.1016/j.infsof.2010.01.001.

Fitzgerald, B., 1997. The use of systems development methodologies in practice: a field study. Inform. Syst. J. 7 (3), 201–212.

Forward, A., Lethbridge, T.C.,2008. Problems and opportunities for model-centric versus code-centric software development: a survey of software professionals. In: Proceedings of the 2008 International Workshop on Models in Software Engineering. MiSE'08. ACM, New York, NY, USA, pp. 27–32, http://dx.doi.org/10.1145/1370731.1370738.

Fowler, M., 1997. UML Distilled: Applying the Standard Object Modeling Language. Addison-Wesley, Boston, MA, USA.

Fowler, M., Scott, K., 2004. UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd ed. Addison-Wesley, Boston, MA, USA.

France, R., Rumpe, B.,2007. Model-driven development of complex software: a research roadmap. In: 2007 Future of Software Engineering. FOSE'07. IEEE Computer Society, Washington, DC, USA, pp. 37–54, http://dx.doi.org/10.1109/FOSE.2007.14.

Fricker, S., Gorschek, T., Byman, C., Schmidle, A., 2010. Handshaking with implementation proposals: negotiating requirements understanding. IEEE Software 27 (March (2)), 72–80, http://dx.doi.org/10.1109/MS.2010.44.

Gliner, J.A., Morgan, G.A., 2000. Methods in Applied Settings: An Integrated Approach to Design and Analysis. Lawrence Erlbaum Associates.

Glück, R., Visser, E., 2011. Special issue on generative programming and component engineering (selected papers from GPCE 2004/2005). Sci. Comput. Program. 76 (5), 347–348 http://www.sciencedirect.com/science/article/pii/S0167642311000220

Gorschek, T., Garre, P., Larsson, S., Wohlin, C., 2006. A model for technology transfer in practice. IEEE Softw. 23 (6), 88–95.

Gorschek, T., Garre, P., Larsson, S.B.M., Wohlin, C., 2007. Industry evaluation of the requirements abstraction model. Requir. Eng. 12 (July (3)), 163–190, http://dx.doi.org/10.1007/s00766-007-0047-z.

Gorschek, T., Tempero, E., Angelis, L., May, 2010. A large-scale empirical study of practitioners' use of object-oriented concepts. In: ACM/IEEE 32nd International Conference on Software Engineering (ICSE), pp. 115–124.

Greenacre, M.J., 1984. Theory and Applications of Correspondence Analysis. Academic Press.

Grossman, M., Aronson, J.E., McCarthy, R.V., 2005. Does UML make the grade? Insights from the software development community. Inform. Softw. Technol. 47 (6), 383–397 http://www.sciencedirect.com/science/article/pii/S095058490400134X

Harel, D., Jan. 1992. Biting the silver bullet: Toward a brighter future for system development. Computer 25 (1), 8–20, http://dx.doi.org/10.1109/2.161283.

Hutchinson, J., Rouncefield, M., Whittle, J.,2011a. Model-driven engineering practices in industry. In: Proceedings of the 33rd International Conference on Software Engineering. ICSE'11. ACM, New York, NY, USA, pp. 633–642.

Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.,2011b. Empirical assessment of mde in industry. In: Proceedings of the 33rd International Conference on Software Engineering. ICSE'11. ACM, New York, NY, USA, pp. 471–480.

IEEE/ACM, 2004. Software Engineering 2004 Degree Programs in Software Engineering. http://sites.computer.org/ccse/

IEEE/ACM, 2009. Graduate Software Engineering 2009 (gswe2009) Curriculum Guidelines for Graduate Degree Programs in Software Engineering. http://www.gswe2009.org

Jackson, M.A., 1975. Principles of Program Design. Academic Press, Inc., Orlando, FL, USA.

Jacobson, I., 1992. Object Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley, Redwood City, CA, USA.

Jensen, R.W., Tonies, C.C., Fletcher, W.I., 1978. A proposed 4-year software engineering curriculum. SIGCSE Bull. 10 (August (3)), 84–92, http://dx.doi.org/10.1145/953028.804240.

Kobryn, C., Jan. 2002. Will UML 2.0 be agile or awkward? Commun. ACM 45 (1), 107–110, http://dx.doi.org/10.1145/502269.502306.

Lauesen, S., 2002. Software Requirements: Styles and Techniques. Addison-Wesley.

Lubars, M., Potts, C., Richter, C.,1993. Developing initial OOA models. In: Proceedings of the 15th International Conference on Software Engineering. ICSE'93. IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 255–264 http://dl.acm.org/citation.cfm?id=257572.257629

Luff, P., Heath, C., Greatbatch, D.,1992. Tasks-in-interaction: paper and screen based documentation in collaborative activity. In: Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work. CSCW'92. ACM, New York, NY, USA, pp. 163–170, http://dx.doi.org/10.1145/143457.143475.

Mohagheghi, P., Dehlen, V., 2008. Where is the proof? A review of experiences from applying MDE in industry. In: Schieferdecker, I., Hartman, A. (Eds.), Model Driven Architecture – Foundations and Applications. Vol. 5095 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 432–443, http://dx.doi.org/10.1007/978-3-540-69100-6_31.

Mohagheghi, P., Gilani, W., Stefanescu, A., Fernandez, M., 2012. An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases. Emp. Softw. Eng., 1–28, http://dx.doi.org/10.1007/s10664-012-9196-x.

Nugroho, A., Chaudron, M.R.,2008. A survey into the rigor of UML use and its perceived impact on quality and productivity. In: Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. ESEM'08. ACM, New York, NY, USA, pp. 90–99, http://dx.doi.org/10.1145/1414004.1414020.

Petre, M.,2009. Insights from expert software design practice. In: Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. ESEC/FSE'09. ACM, New York, NY, USA, pp. 233–242, http://dx.doi.org/10.1145/1595696.1595731.

Petre, M.,2013. Uml in practice. In: Proceedings of the 2013 International Conference on Software Engineering. ICSE'13. IEEE Press, Piscataway, NJ, USA, pp. 722–731 http://dl.acm.org/citation.cfm?id=2486788.2486883

Punter, T., Ciolkowski, M., Freimut, B., John, I.,2003. Conducting on-line surveys in software engineering. In: Proceedings of the 2003 International Symposium on Empirical Software Engineering. ISESE'03. IEEE Comput. Soc., Washington, DC, USA, p. 80 http://dl.acm.org/citation.cfm?id=942801.943614

Riemenschneider, C.K., Hardgrave, B.C., Davis, F.D., Dec.2002. Explaining software developer acceptance of methodologies: a comparison of five theoretical models. IEEE Trans. Softw. Eng. 28 (12), 1135–1145, http://dx.doi.org/10.1109/TSE.2002.1158287.

Robson, C., 2002. Real World Research: A Resource for Social Scientists and Practitioner-Researchers, 2nd ed. Blackwell Publishing.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., 1991. Object-oriented Modeling and Design. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Schauer, R., Keller, R.,1998. Pattern visualization for software comprehension. In: Proceedings of the 6th International Workshop on Program Comprehension. IWPC'98. IEEE Computer Society, Washington, DC, USA, pp. 4–12 http://dl.acm.org/citation.cfm?id=580914.858219

Schmidt, D.C., 2006. Guest editor's introduction: model-driven engineering. Computer 39, 25–31.

Selic, B., 1998. Using UML for modeling complex real-time systems. In: Mueller, F., Bestavros, A. (Eds.), Languages, Compilers, and Tools for Embedded Systems. Vol. 1474 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 250–260, http://dx.doi.org/10.1007/BFb0057795.

Sheskin, D.J., 2011. Handbook of Parametric and Nonparametric Statistical Procedures, 5th ed. Chapman and Hall/CRC.

Sliwa, C., March, 2004. Sidebar: Waiting for UML – 2.0 – Interview with Grady Booch and Bran Selic. Computerworld. http://www.computerworld.com/s/article/91325/Sidebar_Waiting_for_UML_2.0

Stevens, W.P., Myers, G.J., Constantine, L.L., 1974. Structured design. IBM Syst. J. 13 (2), 115–139 http://domino.research.ibm.com/tchjr/journalindex.nsf/495f80c9d0f539778525681e00724804/751ad3c1867f587a85256bfa00685aaa?OpenDocument

Strauss, A.L., Corbin, J.M., 1998. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Vol. 2nd. Sage Publications http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0803959400

Thomson, C., Holcome, M., Cowling, T., Simons, T., Michaelides, G.,2008. A pilot study of comparative customer comprehension between extreme X-machine and UML models. In: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement. ESEM'08. ACM, New York, NY, USA, pp. 270–272, http://dx.doi.org/10.1145/1414004.1414048.

Tomassetti, F., Torchiano, M., Tiso, A., Ricca, F., Reggio, G., may 2012. Maturity of software modelling and model driven engineering: A survey in the italian industry. In: 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012), pp. 91–100.

Torchiano, M., Tomassetti, F., Ricca, F., Tiso, A., Reggio, G., 2011. Preliminary findings from a survey on the MD state of the practice. In: International Symposium on Empirical Software Engineering and Measurement, pp. 372–375.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C.B.R., Wesslén, A., 2000. Experimental Software Engineering – An Introduction. Kluwer Academic Publishers.