A Decade of Software Design and Modeling: A Survey to Uncover Trends of the Practice

Omar Badreddin
University of Texas
El Paso, Texas, U.S.A
obbadreddin@utep.edu

Rahad Khandoker
University of Texas
El Paso, Texas, U.S.A
karahad@miners.utep.edu

Andrew Forward
University of Ottawa
Ottawa, ON, Canada
aforward@uottawa.ca

Omar Masmali
University of Texas
El Paso, Texas, U.S.A
oamasmali@miners.utep.edu

Timothy C. Lethbridge
University of Ottawa
Ottawa, ON, Canada
tcl@eecs.uottawa.ca

ABSTRACT

We present the results of a survey of 228 software practitioners conducted on two phases ten years apart. The goal of the study is to uncover trends in the practice of software design and the adoption patterns of modeling languages such as UML. The first phase was conducted in April-December 2007 and included 113 responses. The second phase was conducted in March-November 2017 and included 115 responses. Both surveys were conducted online, employed identical solicitation mechanisms, and included the same set of questions. The survey results are analyzed within each phase and across phases. We present the results and analysis of the data identifying upward and downward trends in design and modeling practices. The results suggest some increase in formal and informal modeling and identify key challenges with modeling platforms and tools. The results can help researchers, practitioners, and educators to focus efforts on issues of relevance and significance to the profession.

CCS CONCEPTS

• **Software and its engineering → Software design engineering**; **Object oriented frameworks**;

KEYWORDS

Software Design, Software Modeling, UML, Practices, Survey.

1 INTRODUCTION

The adoption of design in the software engineering spheres has been far from uniform. In domains sensitive to deficiencies such as safety-critical systems, software engineers have unreservedly adopted development practices that ensure exceptionally high levels of reliability and security. Those practices include 1) extensive use of model-driven engineering methodologies where models generate all or most of the executable artifacts, and 2) model-driven testing methodologies where many test cases, scenarios, and test oracles are automatically generated and executed to achieve more efficient testing and enhanced coverage. These approaches enable engineers to verify and validate software systems even in absence of complete or executable code and, in many cases, independently of the platform. Models support advanced simulations of software and physical systems, enabling the testing of rare scenarios that may otherwise be too expensive or risky to execute. Model-centered methodologies ensure that potential vulnerabilities in software codes are discovered early and are not obscured by arbitrary code complexities. It has been argued that dissemination of model-centered methodologies would eventually make its way to the main-stream practices [2].

Despite the near consensus on the value-added of model-centered methodologies [14], many studies have exposed fundamental deficiencies in model-centered approaches [23]. Arisholm et al. [3] concluded that the benefits of using modeling in the form of UML diagrams are all but wiped out by the costs associated with maintaining such models. Iivari [9] explored why modeling tools are not used in general in software projects. In their study of organizations that acquired software design technologies and supporting tools, they report that 70 % of modeling tools are never used after being available for one year, 25 % are used by only a single group, and only 5 % become widely used. Usability of modeling tools has been recognized as a possible factor limiting adoption [1][10][11]. Investigations of software engineering and computer science educational programs suggests limited attention to software design concepts, accompanied by a general perception of lack of effectiveness of design notations such as UML [12][13].

This paper focuses on investigating the trends of adoption of software design and modeling in practice. Towards this goal, we designed a survey and collected data on two phases ten years apart. Both phases were conducted online and employed identical solicitations and questions. This paper is organized as follows. In Section 2 we present related work. In Section 3 we present the methodology, effort to minimize bias, and profiling summary data. In Section 4 we present the results of the survey for both phases. The data analysis identifying trends in the practice are presented in Section 5. We present threats to validity in Section 6. Finally, Discussion and Conclusion are presented in Section 7.

MODELS '18, October 14–19, 2018, Copenhagen, Denmark

ACM ISBN 978-1-4503-4949-9/18/10…$15.00
https://doi.org/10.1145/3239372.3239389

2 RELATED WORK

Whittle et. al. conducted a survey of 450 Model Driven Engineering (MDE) practitioners with the aim to understand the extent to which practitioners adopt model driven development styles [15]. Their survey results suggest that MDE adoption may be more widespread than commonly believed. They found that developers often do not use models to generate complete systems. Another survey conducted by Silva aims at identifying key concepts and terminologies in the model driven engineering domain [16]. Their goal is to provide answers for fundamental questions such as; what constitute a model? what is the relationship between a model and a metamodel? and what are the fundamental facets of modeling languages? Another survey study of practitioners focused on practices in Turkey [7]. The authors report significant level of UML and design practices. They also found that waterfall process, despite being old-fashioned, is broadly practiced.

In education, Badreddin et. al. collected 195 student responses from seven programs at four higher education institutions [12]. The goal of the study is to uncover any trends in students' perceptions of the effectiveness of UML in software development. The authors found consistent downward trend in how students perceive UML effectiveness as they progress towards their degree. In a similar study by Liebel that included 218 student responses, the authors found that UML modeling tool complexity to be a significant factor in education [13]. They recommend the use of education specific UML modeling tools to enhance students' learning experiences. In another study that focused on embedded systems, the authors surveyed 275 engineers and found that model driven approaches enhance productivity and the portability of the developed embedded software [21]. We summarize other survey studies of design and modeling practices in Table 1.

3 METHODOLOGY

We present the solicitation approach, survey structure and topics, efforts to minimize bias, and the demographics information of respondents. A complete report on the set of questions, methods, and specific measures taken to minimize threats to validity is published in a separate attached artifact as a technical report.

3.1 Solicitations

The survey was conducted online [8]. We sent targeted solicitations to a wide variety of organizations and posted the survey on many forums, including Javaranch [18], Javaforum [19], Dream in code [20] , several UML user groups, and agile methodology user groups. We posted the survey in technological websites, such as digg.com and dzone.com. We also posted the survey in different Facebook programming groups such as JavaScript, Php Programming, and others. We followed identical solicitation venues, techniques, and frequencies for both phases of the study.

3.2 Survey Structure

The survey consisted of nine topics. Each topic is explored by a set of questions with possible answers varying from a 5-point Likert scale, to open ended free text questions. The Likert scale range is from Strongly Disagree to Strongly Agree, never, always, and Not Applicable. Only complete responses were included in the analysis, but participants were able to skip some questions based on their answers. For example, participants who did not have an experience with a specific approach were able to skip all related sub-questions. In total, the survey included 152 questions. Each survey is expected to take about 60 minutes for completion. The survey topics are as follows.

Topic 1 (fundamentals): This topic constructs a characterization on what constitutes a software design activity and a software model. Various options were presented ranging from class diagrams, use cases, to source code. The objective is to uncover any perceived notions of what constitutes a design or modeling activity.

Topic 2 (basic characterization of the practices): This topic explores the basic characteristics of the practice, including how the modeling and designing activity is accomplished and when, and what notations are used in the course of software design. The objective of this topic is to develop basic understanding of the state of the practice.

Topic 3 (life cycle): This topic investigates the activities involved across various development phases from requirements, design, testing, and documentation. The goal is to characterize the various activities performed throughout the entire software life cycle.

Topic 4 (platform): This topic explores the specific tools, methods, technologies, and platforms used in the software development activities. This topic also documents characteristics of the nature of the software applications that participants develop.

Topic 5 (efficacy): This topic investigates specifics about the design and modeling practices. The questions under this topic explore the various activities and inquires about the efficacy of the various approaches for the task at hand.

Topic 6 (code versus model centrism): This topic explores perceived challenges in code-centric software development approaches and perceived challenges in model-centric development approaches.

Topic 7 (open ended and optional contact information): This topic includes open ended set of questions and comments about the software design practices and questions about the survey itself. This includes optional questions where the participants can voluntarily provide contact information for follow up.

Topic 8 (demographics): Demographics question with sub-questions that include country of origin, education level, and years of experience of the participant.

3.3 Minimization of Bias

To minimize bias, we employed two survey techniques; randomization of questions and balancing positive/negative questions. Randomization was applied to the order of questions to ensure that possible participant fatigue is distributed across survey questions. Specifically, the following question order was randomized: question 2 to question 5 inclusive, and question 7 to question 17 inclusive. To minimize bias in question wording, we adopted neutral question wording whenever possible. In cases where neutral wording was not possible, we balanced the number of positive and negative wording. For example, questions that explore advantages of a specific design and modeling activity is balanced by questions on the advantages of code-centric approaches. Moreover, in multiple steps in the survey, we stated additional information so that participants could consistently answer a set of questions, as an example:

Table 1: Summary of survey studies of design and modeling practices

	Scale/ Region	Year	Student/ Participants	Count	Medium	Goal/focus area	Key Findings
[2]	ABB	2016	Professionals	16	Online Interview	Assessment of UML based development	Evaluation of UML adoption in a specific project
[17]	Global	2007	Professionals	113	Online	Characterization of code-centric versus model-centric development methodologies	Identification of key benefits and challenges with model-centric development
[7]	Turkey	2015	Professionals	202	Online	software engineering practices in Turkey	On average, design related activities consumed 12% of total project effort
[12]	U.S, Canada, Israel	2015	Students	195	Online	Trends in students' perceptions	Students graduate with overall negative perceptions on the effectiveness of UML
[13]	U.S, Canada, Sweden	2017	Students	218	Paper based questionnaire	Case study in Model Driven Engineering Pedagogies	Tools complexity reduces MDE pedagogy effectiveness
[21]	Brazil	2013	Professionals	275	Online	Use of UML modeling and model-driven approaches for embedded software	Model-driven approaches provide productivity and portability of embedded software systems
[22]	Global	2008	Professionals	284	Online	Different dimensions of UML usages	This study suggests several aspects of UML adoption and there is no standard approach to using the UML within a group

"For the remainder of the survey, please assume that any reference to a software model refers to an artifact that represents an abstraction of the software you are building. A model can typically be viewed as a set of diagrams and/or pieces of structured text. It can be recorded on a white board, paper, or using a software tool. A model could use formal syntax and semantics but this is not necessary. We will consider the final source code of the system, and requirements written in natural language to not be models, although models can be embedded in a requirements document." Since the goal of this study is to discover trends, bias inherit in the survey structure and questions will be present in both phases, and therefore will be largely minimized in the course of trends analysis.

3.4 Demographics

The survey collected demographics information related to years of experience, the level of education of participants, their geographical location, and the nature of their professional software engineering role and activities they perform, and the nature of application they develop. Sub-sample analysis is conducted and is reported in the included artifacts and technical report. In Table 2, we present the demographics summary data for Phase one and Phase two separately.

4 RESULTS

We present the results of the survey as follows. All questions under the same topic are combined within the same table whenever possible. For brevity, we only show the combined values for Strongly Agree and Agree, and combined values for Strongly Disagree and Disagree. We also list the mean value for Phase I and Phase II separately. Mean values are calculated by converting the likert scale to a vector of values from 5 corresponding to Strongly Agree, to 1

Table 2: Profiling data

Category	Phase I		Phase II	
	N	%	N	%
All participants (complete responses only)	113	100	115	100
Participants in U.S/Canada	63	55.7	47	41
Participants in EU	13	11.5	3	9.4
Participants outside US/Canada and EU	14	12.5	16	13.8
Participants with >12 years of experience	60	53.47	55	46.9
Participants with PhD	9	8	12	12.1
Participants with Masters	40	35.4	34	36.4
Participants with Bachelor	35	31	12	12.1
Participants without degrees	29	25	23	19.1

corresponding to Strongly Disagree. For each topic, we also show the mean gap, calculated by the difference between the mean value for phase I and Phase II. Statistical significance is shown in bolded and underlined font. The complete raw data, free text, and summary data is included as an additional artifact with this paper.

4.1 Topic 1: Fundamentals

The primary objective of questions in this topic is to ensure that participants have a general agreement on what constitutes a model. This topic included questions about Class Diagrams, Deployment Diagrams, picture by a drawing tool, textual use case, whiteboard drawing, picture by hand, source code, source code comments, and others. The results of this topic are shown in Table 3.

4.2 Topic 2: Characterization of Practices

This topic focuses on uncovering how participants perform their modeling and design activities, and how they learn about various aspects related to modeling. Participants are asked about the methods they use to create models and are provided with choices that include whiteboard drawing, diagramming tool, word processor, word of mouth, hand written material, comments in source code, modeling or CASE tool, drawing software and others. On how participants learn about modeling, the questions focused on the type of artifacts they referred and how they go about their learning activities. The results for this topic is shown in Table 4 and 5. Another set of questions are related to the type of artifacts the developers refer to as shown in Table 6. The last set of questions in this topic explores participants daily activities as shown in Table 7.

4.3 Topic 3: Lifecycle

Questions in this topic focus on modeling and design activities as it relates to the project lifecycle. The first set of questions explore when design activities take place in relation to coding. The choices for this question include before coding, during coding, after coding, and only on request. A second set of questions in this topic explores the various activities that are performed by the participant. These activities included searching, requirements, design, modeling, testing, coding, transfer, develop tests, and documentation. A third set of questions investigates the daily activities of participants. The survey provided choices that participants could choose from and included additional fields where participants could identify additional activities. The choices that were provided include Think about software system, run or attend meetings, explain software design to others, design a software system, lead software project, conduct a search about software system, model a software system, write new code, maintain existing code, fix bugs, perform manual testing, write or maintain requirements, general administration, and write or maintain test scripts. The data for this question is used primarily in sub-sample analysis. Software modeling and various activities during project life cycle are shown in Table 8 and 9.

4.4 Topic 4: Platforms

For the platform topic, we explore the modeling notations that participants use. Some choices that were provided to participants in the survey included UML (any version), UML 2.*, SQL, Structured Design models, UML 1.*, ERD, Well-defined DSL, ROOM / RT for UML, SDL, Formal (e.g. Z, OCL), BPEL, and others. This topic also explores the use of various modeling tools, such as Eclipse, Visual Studio, Rational RSx, Rational XDE, and others. The topic also explores various related technologies and platforms, such as Java, PHP / Perl, ASP.Net, Ruby / Python, C / C++*, and others. Summary results are shown in Table 11.

4.5 Topic 5: Efficacy

This topic presents questions related to the suitability of the modeling tools for the targeted activities, i.e. developing a design, transcribing a design into digital format, generating code where code is accessible and editable, prototyping a design, brainstorming possible designs, generating all code (no manual coding), and others (Table 13). The topic also explores participants' perceptions of key characteristics of modeling tools (i.e. modeling tools suitability as a medium of communication with other developers, ease and speed to create models, suitability for model-based analysis, support for collaborations, visualization of different aspects of the models, generate code, embed parts of models in documents, etc.).

4.6 Topic 6: Code versus Model Centralism

This topic investigates in depth perceptions of participants on when coding approaches versus modeling and design approaches are more suitable to perform various activities. These activities include fixing a bug, creating efficient software, creating a system as quickly as possible, creating a prototype, creating a usable system for end users, modifying a system when requirements change, creating a system that most accurately meets requirements, creating a reusable system, creating a new system, comprehending a system's behavior, explaining a system to others, etc. The answer choices for these questions ranges from Much easier in models, to much easier in code. The summary results of this topic are shown in Tables 14 and 15.

4.7 Open ended and follow up questions

This topic included multiple open-ended questions about modeling and coding, questions about the survey, and optional contact information for follow up questions. Those who provided their contact information were contacted with summaries of the survey results and were provided the option for additional feedback on the survey itself and on the results. The following are itemized summaries of analysis of the text collected in open ended and follow up questions for both survey phases.

- I have taken courses on UML and software design, but the "culture" here has not yet adopted these concepts. They try to use UML, but merely for analysis, not development. We do not yet have case tools or modeling tools.
- In "the real world" it's necessary to cut these models down to the bare basics. Adding too much details to a model takes too much time, and can potentially confuse developers when it comes to implementation.
- Modeling using a tool is good for documenting a model, but otherwise a piece of paper/whiteboard works better and is more flexible.
- Modeling should be used to validate and share your design ideas. If your model works, you can build it too; and others

Table 3: What is a software model?

Responses for Topic 1: What is a software model							
Entity that might be a model	**Phase I**			**Phase II**			**Mean Gap**
	% SA+A	**% SD+D**	**Mean**	**%SA+A**	**%SD+D**	**Mean**	
Class Diagram	88.4	2.7	4.3	87	4.9	4	-0.3
UML Deployment Diagram	77.5	5.4	4.1	72	17.5	3.8	-0.2
Use Case Diagram	82.1	9.8	4	80	13.5	3.8	-0.3
Picture By Drawing Tool	**85.6**	**7.2**	**4**	**62**	**20.3**	**3.5**	**-0.5**
Textual Use Case	**78.8**	**10.6**	**4**	**59**	**18.4**	**3.5**	**-0.5**
Whiteboard Drawing	**78.8**	**8.8**	**3.9**	**63**	**20**	**3.6**	**-0.4**
Picture By Hand	**57.1**	**9.8**	**3.9**	**61**	**13.4**	**3.5**	**-0.4**
Source Code	46.8	38.7	3.2	47	38.7	3.1	-0.1
Source Code Comment	33.9	41.1	2.9	44	39.9	3	0.1

Table 4: Medium and methods of modeling

Topic 2: Medium and methods of modeling							
Medium or methods used to model	**Phase I**			**Phase II**			**Mean Gap**
	% Never&Sometimes	**% Very Often**	**Mean**	**% Never& Sometimes**	**% Very Often**	**Mean**	
Whiteboard drawing	33.3	45.0	3.2	40	57.9	2.9	-0.3
Diagramming tool (e.g. Visio)	42.3	36.9	2.9	43	43.2	2.8	-0.1
Word processor / text	45.5	26.8	2.8	42	55.3	2.7	-0.1
Word of mouth	**42.3**	**27.0**	**2.8**	**54**	**46.1**	**2.4**	**-0.4**
Handwritten material	51.4	22.5	2.6	49	51.3	2.6	0.0
Comments in source code	51.4	21.6	2.5	49	37.8	2.6	0.1
Modeling tool/CASE	58.9	29.5	2.4	55	29.0	2.5	0.1
Drawing software	72.1	12.6	2.1	68	29.0	2.3	0.2

can learn it more easily. Anything more is a waste of time, anything less will cost you more time in the long run.
- There is a time and cost associated with producing the model upfront, but that time is more than gained back during development and, especially, maintenance.
- Now if the [modeling] tools were actually developed by modelers they'd be much better! The tools are often too code centric.
- Code wins over models every time when it comes to revenue. Working and tested code with business value can be sold. Models don't sell, well, unless you are in a huge defense contracting world.
- Model based systems tend to be far more reliable.
- One day there will be no need for learning languages to develop systems.
- Modeling using documentation / plain English is great. Modeling using any formal language or tool has been proven useless for all of my applications.
- First, we have to model the software according to the requirements fulfilling of end user [needs / requirements].
- I love modeling but our team does not have a culture of doing it and it is primary because they do not think we have enough time. Inevitably we waste lots of time because of our lack of models.
- I've been a professional in the industry since 1981. CASE tools were popular and successful in limited domains, but are hard to maintain.
- I have done a fair bit of software modeling but it's been informal. I have no experience with design programs and only academic experience with things like UML.
- Modeling and coding both should work together.

Table 5: What models are used for

Topic 2: What models are used for?							
Activity	**Phase I**			**Phase II**			**Mean Gap**
	% Never & Sometimes	**% Very Often**	**Mean**	**% Never & Sometimes**	**% Very Often**	**Mean**	
Developing a design	26.6	48.4	3.3	28	55.1	3.2	-0.1
Transcribing a design into digital format	32.8	39.1	3.1	41	51.7	2.9	-0.2
Prototyping a design	**53.1**	**32.8**	**2.7**	**24**	**32.2**	**2.2**	**-0.5**
Brainstorming possible designs	**54.7**	**23.4**	**2.6**	**34**	**44.8**	**3**	**0.4**
Generating code (code editable)	65.1	17.5	2.2	66	34.4	2.2	0
Generating all code	76.6	14.1	1.8	66	31	2.1	0.3

Table 6: Reference materials

Responses for Topic 2: Reference materials							
Refer to material created by/as	**Phase I**			**Phase II**			**Mean Gap**
	% Never and Sometimes	**% Very Often**	**Mean**	**% Never and sometimes**	**% Very Often**	**Mean**	
Word of mouth	22.3	54.5	3.4	40	60.5	3.1	-0.3
Word processor / text	30	48.2	3.3	29	54	2.9	-0.4
Diagramming tool	32.4	42.3	3.1	70	36.9	2.7	-0.4
Whiteboard drawing	34.5	41.8	3	37	48.6	2.7	-0.3
Comments in source code	42	30.4	2.9	55	47.3	2.7	-0.2
Drawing software	57.8	13.8	2.6	32	39.5	2.4	-0.2
Modeling tool/CASE	55.9	31.5	2.5	85	28.9	2.3	-0.2
Handwritten material	56	20.2	2.4	27	29.7	2.3	-0.1

5 ANALYSIS

We provide the analysis of the results as follows. We present the upward and downward trends as exhibited in the survey results over the ten-year period. We also present what can be considered as a positive trend and a negative trend. We then summarize the expected and unexpected results. Finally, we present persistent challenges and emerging opportunities.

5.1 Upward and Downward trends

We present analysis of the data based on upward and downward trends as it is manifested in the change between the data set for Phase I and Phase II.

Upward Trends. There is a significant uptake in the use of well-defined and formal modeling languages, such as OCL, as well as well-formed Domain Specific Languages (DSLs). This is also consistent with significantly more participants reporting generating all system code automatically from models (forward engineering), as well as more participants reporting not editing the generated code.

In cases where all code is automatically generated, there is less of a concern about the synchronization between the models and code as the system continues to evolve. This is evident in a significant decrease in participants concerns about model and code becoming out of synchronization. This is also supported by the sub-sample analysis showing high correlation between participants who reported generating all or most of the code and their responses showing little or no concern about code/model synchronization.

Creation of complete fully-executable models often involves a number of modeling notations and languages and requires the specification of multiple aspects of the system under design and development. This is reflected by the results where significantly more participants reported that is it very important that the modeling tool be able to view multiple aspects of the model under development.

Another significant, and related, upward trend is participants reporting that one of their most desired features in modeling tools is an ability to provide high level of information density.

There is also evidence for significant increase in the use of data modeling using ERD models. There are also more participants who reported creating models 'only' upon request.

We are living through a significant flux in middleware, platforms and technologies. This fact seems to be reflected in two ways; 1) significant increase in participants concern about tool providers not continuing to provide support for their modeling tools, and 2) significant increase in concerns related to programming languages and related technologies becoming quickly obsolete. .

Participants reported significant increase in the need for creating reusable designs and systems, and significant increase in using models in brainstorming sessions.

Table 7: Daily activities of participants

Responses for Topic 2: Daily activities of participants							
Available tasks	**Phas I**			**Phase II**			**Mean Gap**
	% Never&Sometimes	**% Very Often**	**Mean**	**% Never& Sometimes**	**% Very Often**	**Mean**	
Think about s/w system	9.4	77.1	4.1	12	41.2	4.1	0
Run / attend meetings	19.8	60.4	3.6	14	68.6	3.5	-0.1
Explain s/w design to others	15.8	51.6	3.5	26	65.7	3.2	-0.3
Design a s/w system	18.8	57.3	3.5	34	54.3	3.3	-0.2
Lead software project	29.2	53.1	3.3	23	65.7	3.2	-0.1
Search about s/w system	31.2	46.2	3.2	31	51.4	3.2	0
Model a s/w system	30.2	45.8	3.2	37	45.8	3.1	-0.1
Write new code	37.5	49	3.1	29	54.3	3.3	0.1
Maintain existing code	37.5	40.6	3	26	60	3.3	0.3
Fix bugs	**39.4**	**39.4**	**3**	**23**	**48.6**	**3.5**	**0.5**
Perform manual testing	35.1	34	2.9	37	51.4	3.1	0.2
Write / maintain requirements	41.1	40	2.9	34	48.6	3.1	0.2
General administration	40.4	29.8	2.8	43	54.3	2.8	0
Write / maintain test scripts	**58.3**	**17.7**	**2.4**	**47**	**44.1**	**2.8**	**0.4**

Table 8: Various activities throughout the project lifecycle

Topic 3: When do you perform the following tasks?					
Available tasks	**Phase I**		**Phase II**		**% Gap**
	Mode	**%**	**Mode**	**%**	
Searching	Constantly	64.5	Constantly	36.1	-28.4
Requirements	Start	60	Start	72.2	12
Design	Start	53.8	Start	44.4	-9.4
Modeling	Start	46.5	Start	66.7	20.2
Perform testing	Constantly	44.1	Constantly	42.9	-1.2
Coding	Constantly	41.7	Constantly	31.4	-10.3
Knowledge transfer	Constantly	41.7	Constantly	30.6	-11.1
Develop tests	Constantly	40.2	Constantly	34.3	-5.9
Documentation	End	38.7	End	27.8	-10.9

Table 9: When is modeling performed?

Topic 3: When is modeling performed?							
Timeline	**Phase I**			**Phase II**			**Mean Gap**
	% Never&Sometimes	**% Very Often**	**Mean**	**%Never & Sometimes**	**% Very Often**	**Mean**	
Before coding	18.8	59.8	3.7	16	54	3.7	0
During coding	33.3	36	3.1	41	51.3	2.8	-0.3
After coding	60.4	19.8	2.5	54	37.8	2.5	0
Only on request	**78.5**	**10.3**	**1.9**	**59**	**32.4**	**2.3**	**0.4**

There also seems to be a broadening in the methods to approach modeling as evident in increase in casual modeling using pen and paper. This is accompanied by participants reporting generally a stricter definition of what constitutes a model (i.e. more participants view that a textual use case is not considered a model).

There is also a positive trend of more participants viewing the generated code as being suitable for their end purposes and its quality matches or exceeds their expectations. There seems to be two factors that may be behind participants' increase satisfaction with the generated code. First, increase in adoption of DSLs often suggests more use of customized generated code, which is more likely to match the developers' particular needs. Second, it is possible that modeling tools in general have improved their code generation.

We investigated the specific modeling tools that were reported in the survey as part of answers to a specific question (i.e. questions under topic 4: Platforms) as well as tools mentioned in the free text and open ended questions. These tools include Papyrus, PlantUML, txtUML, MagicDraw and others. It is possible that code generation improvements in these tools are reflected in participants responses.

Another important trend is the increase in recognition that programming languages and related technologies and platforms could become quickly obsolete. This trend is particularly positive as it is a motivation for adopting model-centric approaches that tend to provide better support for platform independence.

Downward Trends. There is significant decrease in participants satisfaction with the modeling tools they have or are using, as follows. More participants reported that 1) modeling tools are less

Table 10: Modeling notations and tools

Topic 4: Modeling notations and tools							
Modeling notations	**Phase I**			**Phase II**			**Mean Gap**
	% Never&Sometimes	**% Very Often**	**Mean**	**% Never & Sometimes**	**% Very Often**	**Mean**	
UML (any version)	30.9	51.8	3.3	46	33.4	2.9	-0.4
UML 2.*	52.1	34.4	2.6	53	34.4	2.5	-0.1
SQL	55.6	29.6	2.5	49	34.3	2.7	0.2
Structured Design models	58.8	21.6	2.5	50	38.2	2.7	0.2
UML 1.*	**54.8**	**28**	**2.4**	**73**	**26.7**	**1.9**	**-0.5**
ERD	**63.2**	**20.8**	**2.3**	**46**	**40**	**2.9**	**0.6**
Well-defined DSL	**78.8**	**5.8**	**1.7**	**62**	**32.3**	**2.4**	**0.7**
ROOM / RT for UML	85.9	7.1	1.5	79	15.2	1.8	0.3
SDL	**89.2**	**3.2**	**1.3**	**68**	**25.8**	**2.2**	**0.9**
Formal (e.g. Z, OCL)	**93.9**	**2**	**1.3**	**75**	**18.8**	**1.9**	**0.6**
BPEL	92.8	3.1	1.3	87	13	1.6	0.3

Table 11: Technology

Technology options	Phase I			Phase II			Mean Gap
	% Never& Sometimes	% Very Often	Mean	%Never & Sometimes	% Very Often	Mean	
Java	**46.3**	**31.6**	**2.4**	**80**	**11.5**	**1.8**	**-0.6**
PHP / Perl	74.2	19.4	2	74	14.3	2.2	0.2
ASP.Net	79.4	14.4	1.8	74	14.3	2	0.2
Ruby / Python	**88.3**	**8.5**	**1.6**	**77**	**17.2**	**1.9**	**0.3**
C / C++*	60	30	2.4	65	25	2.3	-0.1

capable of supporting communications with other developers and designers; 2) there is inadequate support for maintaining code and model in sync; 3) there is inadequate support for prototyping, and 4) modeling tools are not suitable for creating software designs in general. This is also reflected in a lower frequency of using and reusing of models created by other developers.

There is a decrease in participants' satisfaction with modeling tools as evident by the data showing more participants find modeling tools to be overly complex, to require significant learning curve and to be difficult to use. These issues are also reflected in the open-ended survey questions.

Eclipse, the open source development platform, demonstrated significant decline in use by the survey participants. UML version 1.* has decreased significantly, as to be expected due to more participants using the newer UML versions.

A large majority of the negative trends seems to be related to the perception of modeling tools in terms of usability and suitability for performing many tasks and activities. More participants find modeling tools increasingly difficult to learn and report the learning curve to be a significant challenge.

There is also a general decline in perception of modeling tools support for activities that involve communication and collaboration with others. This may explain why more participants reported creating models upon request only and often using a pen and paper and whiteboards as a modeling platform. Moreover, increasing number of responses indicate less re-use of existing models. This dissatisfaction with modeling tools is repeatedly mentioned in the free text responses. Many participants argued that time investments in model creation and maintenance is not justified. Others argued that casual informal modeling (often without using a modeling tool) is much more effective.

The data also shows a trend of declining number of participants reporting performing the task of transcribing a model from an informal source (such as the whiteboard) to a modeling tool. This could be interpreted as both positive and negative. Positive interpretation is that more participants are creating the models on a modeling tools, and hence not requiring the transcribing task. Alternatively, this trend can be interpreted negatively that participants do not find the modeling tool flexible enough to support their need for sketching and creating models in an agile and quick fashion.

5.2 Expected and Unexpected Trends

The primary expected trend is the decline in the use of older UML versions.

The first unexpected result is increase in the practices of modeling and design despite the decrease in participants satisfaction with modeling tools. We also did not expect that participants satisfaction with modeling tools to decline significantly. Another unexpected result is related to the increase in the adoption of formal modeling and domain specific modeling languages (DSLs) even though we sought participants from development venues where we wouldn't expect that DSLs be broadly adopted.

5.3 Persistent Challenges and Emerging Opportunities

We identify persistent challenges as those issues in modeling and design that are found to be problematic in both phases with little or no improvement in between the two data sets. Many of these

Table 12: Modeling Tools

What are the desired attributes of a modeling tool?							
Modeling tools attributes	Phase I			Phase II			Mean Gap
	Rank	Mean	SD	Rank	Mean	SD	
Communicate to others	**1**	**5**	**4**	**1**	**4.2**	**4**	**-0.8**
Readability	**2**	**5**	**4**	**2**	**4.5**	**3**	**-0.5**
Ease and speed to create	3	3	2	3	3.3	3	0.3
Ability to analyze	4	3	3	4	3.1	3	0.1
Collaborate amongst developers	5	3	2	5	4	3	1
Ability to view different aspects of a model	**6**	**2.3**	**2**	**6**	**3**	**3**	**0.7**
Generate code	7	3	2	7	3	2	0
Information density	**8**	**3.2**	**2**	**8**	**3.9**	**3**	**0.7**
Embed parts of model in documentation	9	3	2	9	2.8	2	-0.2

Table 13: Modeling tools good at

How good are modeling tools for ?							
Available activities	**Phase I**			**Phase II**			**Mean Gap**
	% Poor	**% Good**	**Mean**	**% Poor**	**% Good**	**Mean**	
Developing a design	**16.9**	**47.9**	**3.4**	**11**	**53.6**	**2.9**	**-0.5**
Transcribing a design into digital format	24.6	42	3.2	25	60.7	3.3	0.1
Generating code (code is editable)	39.1	29	2.9	32	64.3	3	0.1
Prototyping a design	41.2	29.4	2.9	25	71.4	3.1	0.2
Brainstorming possible designs	45.1	32.4	2.8	18	74.7	3.1	0.3
Generating all code (no manual coding)	**79.7**	**8.7**	**1.9**	**50**	**42.9**	**2.5**	**0.6**

Table 14: Tasks are better in a model centric versus code centric approach

Topic 6: Available activities	**Phase I**			**Phase II**			**Mean Gap**
	% Easier in Models	**% Easier in Code**	**Mean**	**% Easier in Models**	**% Easier in Code**	**Mean**	
Fixing a bug	28.9	43.3	3.2	19	40.6	3.2	0
Creating efficient software	35.9	43.5	3.1	27	50	3.2	0.1
Creating a system as quickly as possible	46.7	42.4	3	31	56.2	3.2	0.2
Creating a prototype	43	32.6	2.9	44	37.5	2.7	-0.2
Creating a usable system for end users	42.4	22.8	2.7	49	27.3	2.4	-0.3
Modifying a system when requirements change	54.9	24.2	2.5	41	37.5	2.8	0.3
Creating a system that most accurately meets requirements	67	19.8	2.2	56	26.4	2.3	0.1
Creating a re-usable system	**63**	**15.2**	**2.2**	**42**	**30.4**	**2.6**	**0.4**
Creating a new system overall	68.5	20.7	2.2	64	24.2	2.3	0.1
Comprehending a system's behaviour	71.9	15.7	2	75	15.7	1.9	-0.1
Explaining a system to others	81.8	7.6	1.7	66	15.6	1.9	0.2

challenges manifest themselves in modeling tools complexity, their required learning curve and their inadequate support for flexibility.

Another persistent challenge is related to the practice of developing executable models or relying on models to generate all code and executable artifacts. This challenge, despite being slightly reduced, remains a significant limiting factor for broader adoption. The need to modify the generated code because it is incomplete, or because of the need to add functionality introduces a whole set of challenges, including managing the synchronization between code and models.

A third challenge is the limited scope of modeling activities in terms of their occurrences in the project lifecycle, or along the various activity disciplines. It is only in the requirements phase that more than half of the participants reported using modeling frequently. Across other activities, modeling remains rather low. Moreover, almost all of potential problems with model-centric approaches identified by participants has remained or declined only slightly.

The key emerging opportunity is the significant increase in modeling activities in general, and particularly the increase in the adoption of Domain Specific Modeling Languages and formal modeling.

6 THREATS TO VALIDITY

The primary threats to validity of this study are summarized below. We have also outlined the steps we have taken to help mitigate these threats.

Question interpretation. Respondents may have misunderstood the intended meaning of our questions. We took two steps to reduce the ambiguity of the questions. First, five independent

Table 15: Problems with model centric approach

Topic 6: Problems with Model-Centric Approaches	**Phase I**			**Phase II**			**Mean Gap**
	% Slight Problem	**% Bad Problem**	**Mean**	**% Slight Problem**	**% Bad Problem**	**Mean**	
Models become out of date and inconsistent with code	**16.3**	**68.5**	**3.8**	**25**	**40.6**	**3.2**	**-0.6**
Models can not be easily exchanged between tools	26.4	51.6	3.3	19	40.7	3.3	0
Modeling tools are 'heavyweight'(install,learn,configure,use)	31.5	39.1	3.1	41	37.6	3	-0.1
Code generated from modeling tool not of the kind kind I would like	39.6	38.5	3	44	31.3	2.7	-0.3
Cannot model in enough detail-must write code	43.8	36	2.8	47	28.1	2.6	-0.2
Creating and editing model is slow	43.5	22.8	2.7	38	34.4	3	0.3
Modeling tools change, models become obsolete	44.6	32.6	2.7	31	34.4	3	0.3
Modeling tools lack features I need or want	44.9	21.3	2.6	44	18.8	2.6	0
Modeling tools hide too many details(fully visible in source)	44.6	23.9	2.6	34	31.3	2.9	0.3
Modeling tools are too expensive	46.7	26.7	2.6	38	15.7	2.7	0.1
Modeling tools cannot be analyzed as intended	51.1	25.6	2.5	56	21.9	2.5	0
Semantics of models different from prog. language	56.7	23.3	2.4	48	16.2	2.5	0.1
Modeling languages are not expressive enough	54.9	17.6	2.4	50	15.7	2.5	0.1
Modeling languages are hard to understand	62.6	9.9	2.2	58	15.2	2.3	0.1
Have had bad experience with modeling	63.7	16.5	2.2	61	16.2	2.2	0
Do not trust companies will continue to support their tools	**67.4**	**10.1**	**2**	**41**	**15.7**	**2.6**	**0.6**

Table 16: Problems with code centric approach

Topic 6: Problems with Code-Centric Approaches							
Potential problems	**Phase I**			**Phase II**			**Mean Gap**
	% Slight Problem	**% Bad Problem**	**Mean**	**% Slight Problem**	**% Bad Problem**	**Mean**	
Hard to see overall design	13.8	66	3.8	12	67.7	3.8	0
Hard to understand behaviour of system	19.1	60.6	3.6	30	45.4	3.2	-0.4
Code becomes of poorer quality over time	28.3	55.4	3.4	34	43.8	3.2	-0.2
Too difficult to restructure system when needed	22.6	51.6	3.4	22	53.1	3.4	0
Difficult to change code without adding bugs	**22.6**	**50.5**	**3.4**	**38**	**40.6**	**2.9**	**-0.5**
Changing code takes too much time	39.4	27.7	2.8	44	21.9	2.6	-0.2
Our prog. language leads to complex code	51.1	20.2	2.5	41	28.1	2.7	0.2
More skill than available to develop high quality code	53.8	22	2.5	47	31.3	2.7	0.2
Prog. Languages not expressive enough	64.8	14.3	2.1	69	18.8	1.9	-0.2
Organization culture does not like code-centric	72.8	14.1	1.9	78	6.3	1.7	-0.2
Our prog. language likely to become obsolete	**75.3**	**9.7**	**1.9**	**56**	**28.1**	**2.4**	**0.5**

researchers reviewed the survey structure, wording, and questions. Second, we prototyped the survey and reviewed any ambiguities and implemented the suggested comments. Both activities helped improved the overall survey prior to go-live. Since the primary goal of this study is to uncover trends, we expect that bias inherit in the survey to be present in both phases and its effects would be largely minimized.

Researcher bias. Many of the survey questions attempt to uncover trends related to both model-centric and code-centric approaches. A potential bias could be introduced if our survey appeared to be overly negative towards either modeling or software coding. To reduce the chance of bias we aimed to be objective whenever we referring to both code-centric and model-centric questions, as well as presenting the questions in a random order. We also maintained the same questions and wording for the two phases to minimize the effects of any potential bias. We also prototyped the survey and collected feedback from the participants. We revised the wording to ensure consistent interpretation of questions.

Non randomized sample and representation. To help ensure that our sample was based on a representative collection of software practitioners, we approached both open and closed forums for participation. In particular, we submitted survey link to Digg.com, and Dzone.com - two popular technology and news sites. We submitted email requests to UML user groups, agile user groups, Java user groups, and process user groups. Our demographics results indicate that we do have representation from most regions of the world, most educational backgrounds, most software industries, and most types of developers. We also conducted extensive subsample analysis that included analyzing sub-samples of participants, their educational and experience profiles, as well as sub-sample analysis of software application domains to ensure adequate representation.

Participant Fatigue. This survey study included about 152 questions many with multiple choices and open-ended questions with free text. We estimate that the survey takes about one hour to fully answer all questions. Hence, there is a risk of participant fatigue which may affect data validity. We did the following to minimize this risk. First, the survey and our solicitations clearly stated the expected duration of the survey. Second, we included many cues in the survey to inform the participant of progress and the remaining sections. Third, we allowed participants to skip sub-questions based on their responses. Fourth, we only included complete responses in the data analysis. Lastly, end-of-survey free text questions collected feedback including comments about the survey itself. We also collected additional free text in the post-survey phase.

7 CONCLUSION

This paper reports on a survey conducted on two phases ten years apart. The goal of the survey is to discover trends in the practices of software design and modeling. The survey solicited 228 participants and included 152 questions.

The survey analysis characterizes trends in the practice, including upward, downward, positive, negative, and unexpected trends. The survey suggests some level of increase in the adoption of the broad practices of modeling as well as an increase in the adoption of domain specific and formal modeling. We also observed an increase in the practices of casual modeling using whiteboards and pen and paper approaches. The data also suggests a persistent dissatisfaction with software modeling tools. Participants find modeling tools to be inadequate in their support for collaboration and communication.

Participants also consistently reported inadequacy of the generated code for their development purposes and needs.

Many participants argued that modeling cannot be justified due to learning curve, modeling tool complexity, and inadequacy for delivering executable artifacts.

The two phases of the survey draw a picture of the practices and how they are changing over time. Our goal is to help researchers, educators and practitioners to focus on important aspects of relevance to the profession.

REFERENCES

[1] Agarwal, R. and Sinha, A.P.,2003 "Object-oriented modeling with UML: a study of developers' perceptions." *Communications of the ACM.* 46(9), pp.248-256.

[2] Anda, Bente, et al.. "Experiences from introducing UML-based development in a large safety-critical project." *Empirical Software Engineering.* 11.4 (2006): 555-581.

[3] Arisholm, Erik, et al.. "The impact of UML documentation on software maintenance: An experimental evaluation." *IEEE Transactions on Software Engineering* . 32.6 (2006): 365-381.

[4] Hertel, Guido, Sven Niedner, and Stefanie Herrmann."Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel." *Research policy 32.7 (2003).* : 1159-1177.

[5] Procaccino, J. Drew, et al. "What do software practitioners really think about project success: an exploratory study." . *Journal of Systems and Software* . 78.2 (2005): 194-203.

[6] Chow, Tsun, and Dac-Buu Cao. "A survey study of critical success factors in agile software projects." *Journal of systems and software* 81.6 (2008): 961-971.

[7] Garousi, Vahid, et al. "A survey of software engineering practices in Turkey." *Journal of Systems and Software* 108 (2015): 148-177.
[8] Rahad Khandoker and Omar Badreddin "Professional coding and modeling practices, 2017. *Available: https://goo.gl/bQV9Ph.*
[9] Iivari, J. "Why are CASE Tools Not used? " *Communications of the ACM 1996. Communications of the ACM* , vol 39, pp. 94-103".
[10] Agarwal, R., De, P., Sinha, A. P., and Tanniru, M, 2000. "On the usability of OO representations.âĂİ *Communications of the ACM* . 43, no. 10 (2000): 83-89.
[11] Maged Elaasar, Florian Noyrit, Omar Badreddin, SÃľbastien GÃľrard. "Reducing UML Modeling Tool Complexity with Architectural Contexts and Viewpoints". *International Conference on Model-Driven Engineering and Software Development (MODELSWARD)* .2017.
[12] Omar Badreddin, Arnon Sturm, Abdelwahab Hamou-Lhadj, Timothy Lethbridge, Waylon Dixon, Ryan Simmons. The Effects of Education on Students Perception of Modeling in Software Engineering. *In proceedings of ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems.*
[13] Grischa Liebel, Omar Badreddin and Rogardt Heldal. âĂIJModel Driven Software Engineering in Education: A Multi-Case Study on Perception of Tools and UML". *In IEEE Conference on Software Engineering Education and Training (CSEE&T).*
[14] Budgen, David, Andy J. Burn, O. Pearl Brereton, Barbara A. Kitchenham, and Rialette Pretorius. "Empirical evidence about the UML: a systematic literature review." *Software: Practice and Experience.* 41, no. 4 (2011): 363-392.
[15] Whittle, Jon, John Hutchinson, and Mark Rouncefield. "The state of practice in model-driven engineering." *IEEE software 31*, no. 3 (2014): 79-85.
[16] Da Silva, Alberto Rodrigues. "Model-driven engineering: A survey supported by the unified conceptual model." *Computer Languages, Systems & Structures* 43 (2015): 139-155.
[17] Forward, Andrew, and Timothy C. Lethbridge. "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*, pp. 27-32. ACM, 2008.
[18] Online Programming forum JavaRanch. Available: https://javaranch.com/
[19] Online Programming forum JavaForum. Available: https://www.java-forums.org/forum.php
[20] Online Programming forum Dreamincode. Available: http://www.dreamincode.net/forums/
[21] Agner, Luciane Telinski Wiedermann, et al. "A Brazilian survey on UML and model-driven practices for embedded software development." *Journal of Systems and Software.* 86.4 (2013): 997-1005.
[22] Dobing, Brian, and Jeffrey Parsons. "Dimensions of UML diagram use: a survey of practitioners." *Journal of Database Management.* 19.1 (2008): 1
[23] Badreddin, Omar, Timothy C. Lethbridge, and Maged Elassar. "Modeling practices in open source software." In IFIP International Conference on Open Source Systems, pp. 127-139. Springer, Berlin, Heidelberg, 2013.