View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by OpenKnowledge@NAU

The Effects of Education on Students' Perception of Modeling in Software Engineering

Omar Badreddin
Northern Arizona University
Flagstaff, U.S.A
Omar.Badreddin@nau.edu

Arnon Sturm
Ben-Gurion University of the Negev
Beer Sheva, Israel
sturm@bgu.ac.il

Abdelwahab Hamou-Lhadj
Concordia University
Montreal QC, Canada
abdelw@ece.concordia.ca

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

Waylon Dixon
Northern Arizona University
Flagstaff, U.S.A
waylon.dixon@nau.edu

Ryan Simmons
Northern Arizona University
Flagstaff, U.S.A
rsimmons07@gmail.com

***Abstract*— Models in software engineering bring significant potential in improvements of productivity of engineers, and improved quality of the artifacts they produce. Despite this significant potential, modeling adoption in practice remains rather low. Computer Science and software engineering curriculums may be one factor that causes this low adoption.**

In this study, we investigate the effects of education on students' perception of modeling. We conducted a survey in three separate institutions, in Canada, Israel, and the U.S. The survey covers various aspects of modeling and addresses students ranging from a first year in undergraduate studies until final years in graduate studies.

The survey's findings suggest that the perception of undergraduate students towards modeling declines as they progress in their studies. While graduate students tend to be more favorable of modeling, their perception also declines over the years. The results also suggest that students prefer more modeling content to be integrated earlier in the curriculum.

***Index Terms*—Survey of Perceptions, Pedagogy, Modeling in Software Engineering, UML, Education.**

I. Introduction

Model Driven Engineering promotes the use of models, rather than code, for system development. Models can be easier to understand [1][2], improve communications amongst stakeholders [3], and help generate executable artifacts [4]. In addition, platform independent models can improve system portability, and can facilitate migrating systems from legacy platforms [15].

UML has emerged as the standard modeling language in software engineering. In an empirical assessment of MDE in industry, Hutchison et al. [10] mentioned that UML was used by 85 percent of the respondents. Petre [16] mentioned many studies that have indicated the UML is the de facto standard modeling language or the "lingua franca" one. The standard is managed by OMG, and supports many aspects of the life cycle of software development, from requirements, specification, to development and deployment. However, the adoption of modeling in software engineering practice remains dismal. Studies point to significantly low adoption of modeling in practice [2] and that open source projects remain code-centric by and large [5]. Petre [16] also provided evidence that the actual adoption of UML is quite low.

The reasons behind such low adoption may be attributed to many factors. These include complexity of modeling tools, the lack of compatibility within different tools, the lack of integration of modeling tools within existing environments, and the lack of education about the value of modeling tools and techniques [6].

Our goal in this study is to investigate the effect of education in software engineering and computer science on students' perception of modeling. In particular, we want to understand what effect, if any, education has on how students perceive the value of models.

We designed and distributed a survey covering many aspects of modeling for students throughout the full academic spectrum, from undergraduate to post graduate students. The survey is conducted in three separate universities to allow for different cultural and perspectives.

This paper is organized as follows. In Section II, we review the related work. In Section III, we introduce the study design. We then briefly introduce a background on the modeling-related curriculum at the three participating institutions. The results of the survey are presented in section V and further discussed and analyzed the results in Section VI. Finally, In Section VII we conclude and as set plans for future research directions.

II. Related Work

The perception of UML by professional software engineers has been investigated, with mixed results. Ariadi and Chaudron have surveyed 80 professional software engineers about their perception of the value of UML in terms of productivity and quality [7]. Despite the low adoption of UML, its value is perceived very positively in design, analysis and implementation. Other such surveys have reported negative perceptions of UML due to its complexity, incompleteness, and finds that UML is perceived to be difficult to learn [13].

UML as the standard modeling language is increasingly becoming integrated into academic curriculums for undergraduate and graduate students. There has been a number of studies that reported on experiences on teaching UML [12], as well as studies in innovative tools for UML education [11]. In addition, a number of studies on the effectiveness of a specific teaching technique for software engineering students have been reported, such as case study and problem based approaches [8].

There has been a number of studies on the comprehension of specific modeling notation, such as the work of Glezer et al. [9] on the comprehension of UML Interaction diagrams. However, there has been very little known about students' perception of modeling, modeling tools and the curriculum in terms of modeling coverage and depth.

III. Study Design

A. Goal

The goal of this study is to uncover the perceived value and usefulness of models by undergraduate and graduate students in Computer Science and Software Engineering as they progress in their studies. The focus in this study is not on the specific modeling language (e.g., UML or BPMN) but rather on the applicability and usability of models in general.

The research questions we were interested in are the following:

- Do students perceive models useful? And in what context? What are the reasons for that?
- Do students think or wish to have a more substantial modeling education?
- Does students' perception over modeling evolve over their studies?

B. Intended Subjects

The intended subjects of this study are undergraduate and graduate students in system development related fields such as software engineering, computer science, and information systems engineering.

C. Administering the Survey

The survey was filled by students either in classrooms, labs, or online. Participation was both anonymous and voluntary. The survey was conducted in three institutions, Concordia University in Canada, Ben-Gurion University of the Negev in Israel, and in the Northern Arizona University in the U.S.

D. The Survey

The survey consisted of two parts: *demographic data* and a *reflection on modeling*.

The demographic data part included questions regarding the university, the study program, the academic year, age, work experience (with ranges: 0-3, 4-7, 8-12, 13+), and the average grade (with ranges: 65-75, 76-85, 86-90, 91+).

The reflection part included the following questions:

1. Applicability of Models (APP)
   a. Models are very useful
   b. Models are useful for documentation
   c. Models are useful for communication
   d. Models are useful for representing requirements
   e. Models are useful for specification
   f. Models are useful for implementation and/or code generation
   g. Models are useful for testing
   h. Models are useful for maintenance
2. Modeling Characteristics (CHR)
   a. Models are normally used just as drawings
   b. Code is just a type of model
   c. Models are precise (i.e., unambiguous)
   d. Models can be easily checked to find opportunities for improvement
   e. Models are more comprehensible than code
   f. In general, models are easy to understand
   g. Models facilitate abstractions and comprehension
   h. Textual models are easier to understand than graphical models
   i. Textual models are easier to construct than graphical models
   j. Models are implementation independent
   k. Models help provide flexibility during the development process
   l. Modeling is counterproductive since the models need to be changed all the time
   m. Models are usually abandoned after the code is written
3. Implementation (IMP)
   a. Modeling tools are not mature enough
   b. Modeling tools are too complex and are difficult to learn
   c. It is not easy for developers to obtain modeling tools that meet their needs
4. Modeling Education (EDU)
   a. Modeling should be taught before programming
   b. Modeling and programming should be taught at the same time
   c. Modeling is not being taught sufficiently
   d. Modeling should be integrated in most software engineering and computer science courses

These questions had Likert scale: Strongly Agree, Agree, Neutral, Disagree, Strongly Disagree, and NA; representing a scale from 5 to 1. The full list of questions as well as the raw data is included in the supplementary material.

IV. Background on the Institutions and their Modeling Courses

In the following we elaborate on the education background of the participants by introducing the curriculum in each of the institutes.

A. Concordia University

Concordia University (CU) offers two related programs that are both managed by Department of Computer Science and Software Engineering: Computer Science and Software Engineering. The Computer Science program focuses primarily on the study and design of computer systems, such as design algorithms, languages, hardware architecture, systems software, and applications software and tools. Whereas, the

Software Engineering program, while built on the fundamentals of computer science, is focused more on the principles and practices of engineering to develop creative applications such as, computer games, web services, information security, and avionics.

Tables 1 and 2 present the courses in which modeling education takes place along with their modeling content, in both programs. Although two programs are administratively separate, in this survey we unified the results of the two programs since the modeling content is largely similar.

Table 1. Computer Science @ Concordia

Sem	Course	Credit (/90)	Modeling Content
2	Fundamentals of Programming	3	Includes basics of object-oriented programming, essentially UML aspects.
3	System Hardware	3	Abstraction and modeling of system architecture.
4	Object-Oriented Programming 1 & 2	7	Essentially, a modeling course related to all UML aspects.
5	Introduction to Database Applications	4	Modeling DB using ERD
6	Computer Architecture	3	Using modeling construct to teach CA content such as content/data flow, shared memory models, etc.
6-8	Databases	4	Modeling DB using ERD, OOD, and ODL.
6-8	Introduction to Software Engineering	4	Using modeling construct to teach other SE content such as design patterns and refactoring.
6-8	Database Design	4	Essentially, a modeling course related to all UML aspects.
7,8	Computer Science Project 1 & 2	6	Using models to implement and manage a whole project.

Table 2. Software Engineering @ Concordia

Sem	Course	Credit (/120)	Modeling Content
6-8	Software Process	3	Basic principles of SE with activities in software notations and documentations.
6-8	Software Architecture and Design 1 & 2	6	Essentially, a modeling course related to all UML aspects.
6-8	User Interface Design	3	Using modeling construct to teach other UID content such as usability engineering, user models, and notations.
6-8	Control Systems and Applications	3	Using modeling construct to teach CSA content such as block diagrams.
7	Software Engineering Team Design Project	3.5	Using models to implement and manage a software project.
8	Capstone Software Engineering Design Project	4	Using models to implement and manage a whole project.

B. Ben-Gurion University of the Negev

In Ben-Gurion University of the Negev (BGU) there are two system engineering programs which are related to the goal of the survey. The first is the Information System Engineering program in which the focus on data analysis, yet, graduates of that program are expected to perform development activities as well. The program is managed by the department of the Information System Engineering. Table 3 presents the courses that cover software modeling. Other courses refer to information systems management such as production management, organizational culture, information retrieval and data mining, operational research etc. The second program is the Software Engineering, which is managed by the two departments of Information System Engineering and Computer Science. Graduates of that program serve mainly in development positions. Table 4 presents the courses in which modelling education takes place along with their modeling content. Other courses include computer science foundation such as principles of programming languages, automata, compilation, etc.

Table 3. Information Systems Engineering @ BGU

Sem	Course	Scope (/160)	Modeling Content
1	Introduction to Information Systems Engineering	3	Basics of UML, mainly class diagram
5	Database Systems	3.5	Modeling DB using ERD
5	Analysis and Design of Information Systems	5	Focus mainly of functional modeling
6	Object-Oriented Analysis and Design	3.5	Essentially, a modeling course related to all UML aspects.
7-8	Capstone Project	8	Using models to implement and manage a whole project.

Table 4. Software Engineering @ BGU

Sem	Course	Credit (/160)	Modeling Content
2	Introduction to Software Engineering	2.5	Basics of UML, mainly class diagram
3	Database Systems	3.5	Modeling DB using ERD
4	Analysis and Design of Software System	5	Essentially, a modeling course related to all UML aspects as well as DFD.
5	Topics in Software Engineering	4.5	Using modeling construct to teach other SE content such as design patterns, and refactoring.
6	Software Implementation Workshop	3	Using models to implement iteratively a small scale application.
7-8	Capstone Project	8	Using models to implement and manage a whole project.

C. University of Northern Arizona

At Northern Arizona University (NAU), there are two related programs; the first is Computer Science where there is emphasis on theoretical foundation of computer science (Automata theory, Algorithms, etc.) and the second is Applied Computer Science where students are given the option to replace theory courses with more applied courses (such as mobile and web development courses). The Computer Science program at NAU is accredited under ABET [14]. Table 5 presents the courses where modeling is covered along with their content.

Table 5. Computer Science @ NAU

Sem	Course	Credit (/~120)	Modeling Content
1	Introduction to Computer Science I (+lab)	4	Basic Class Diagrams

2	Introduction to Computer Science II (+lab)	4	Basic Class Diagrams
5	Data Base Systems	3	ER Diagrams
6	Software Engineering	3	Many UML notations are presented (class, state machines, use cases)
7	Requirements Engineering (Capstone I)	2	Project-specific UML.

V. Results

In this section, we present the summarized results for each institution. The complete raw data as well as summarized data are made publicly available[1] to facilitate replication and validation of the results [17].

A. Demographics

All in all we got 195 filled questionnaires for the three universities. Analyzing the profiles of the participating students, as they appear in the following tables, we found out that most of the participants have good grades and they have limited work experience (a fact that emphasizes that their perception is mainly established by their education).

Table 6. Profile of the participating students

(a) Number of Responses

Institute	Number of Responses				
	Y1	Y2	Y3	Y4	Grad
NAU	5	10	26	8	2
BGU-SE	8	12	17	22	25
BGU-ISE	3	3	23	12	
CU	0	0	0	0	19

(b) Years of experience

Institute	Experience (years)			
	0-3	4-7	8-12	13+
NAU	46	5	1	0
BGU SE	47	0	1	0
BGU ISE	35	3	0	0
CU	12	5	1	1

(c) Average grades obtained in modeling-related courses

Institute	Average Grade (%)			
	65-75	76-85	86-90	91+
NAU	0	20	14	17
BGU SE	6	25	9	6
BGU ISE	5	23	6	4
CU	0	4	8	7

The response rates we had for the questionnaire are as follows. For BGU-ISE and BGU-SE, as the survey was conducted on line, we got response rate of 13 percent. For BGU graduate students, for NAU, and CU we had a response rate of above 90 percent as the survey was conducted in class as a paper questionnaire.

[1] https://zenodo.org/record/20367?ln=en#.Veuv5dNViFI

B. Reflection on Modeling

Figures 1-3 summarize the average results for each institution and give an overview of the results per institution. In the following, we discuss these results.

In general, the perception of BGU's students towards modeling is positive. In particular, they perceive modeling as a useful means mainly for documentation and communication. One of the reasons for that limited usefulness might be the students' perception of modeling characteristics. For example, the students perceive models as drawings, they find it counterproductive, and they find textual models (like code) easier to deal with. As for the training they receive, the students think that more training on modeling is required.

Overall, the perception of NAU's student of modeling is generally positive. Graduate students seem to appreciate modeling for documentation and communication. But they do not find models to be that useful for representing requirements or for specification. Their perception of models tends to get significantly lower when it comes to using models for testing and maintenance.

NAU Undergraduate students seem to find models to be more comprehensible than code. This could be interpreted by the fact that undergraduates find code to be challenging and/or complex. Graduate students, on the other hand, do not find models to be more comprehensible.

CU students find modeling very useful and that it should be integrated in the curriculum earlier (as shown in Figure 3). They also believe that modeling is important for various software engineering tasks, and not just used for drawing diagrams. Concordia students also believe that textual modeling is not easier to understand and construct than graphical modeling. We attribute this positive reaction to modeling of Concordia students mainly to the fact that they are graduate students. Many of them had taken some graduate courses on modeling as well. It is also interesting to note that when asked whether modeling and programming should be taught at the same time, Concordia students seem less favorable to this idea (average score is 3.4/5). This might indicate that students wish to see more courses dedicated to using models as the main development artifacts. Courses that combine tightly both perspectives (code and models) seem to reiterate the traditional perception of modeling, which restricts models to the design phase only.

The survey's results towards the educational section tend to have an upward trend that is more evident for the graduate students. Students in general want more training in modeling and feel that modeling should be taught at the same time as coding. Interestingly, graduate students tend to agree more. This can be interpreted by the fact that graduates appreciate models more, and have more appreciation on the role of modeling in Software Engineering, and therefore, are more positive regarding increasing the modeling portions in the curriculum.

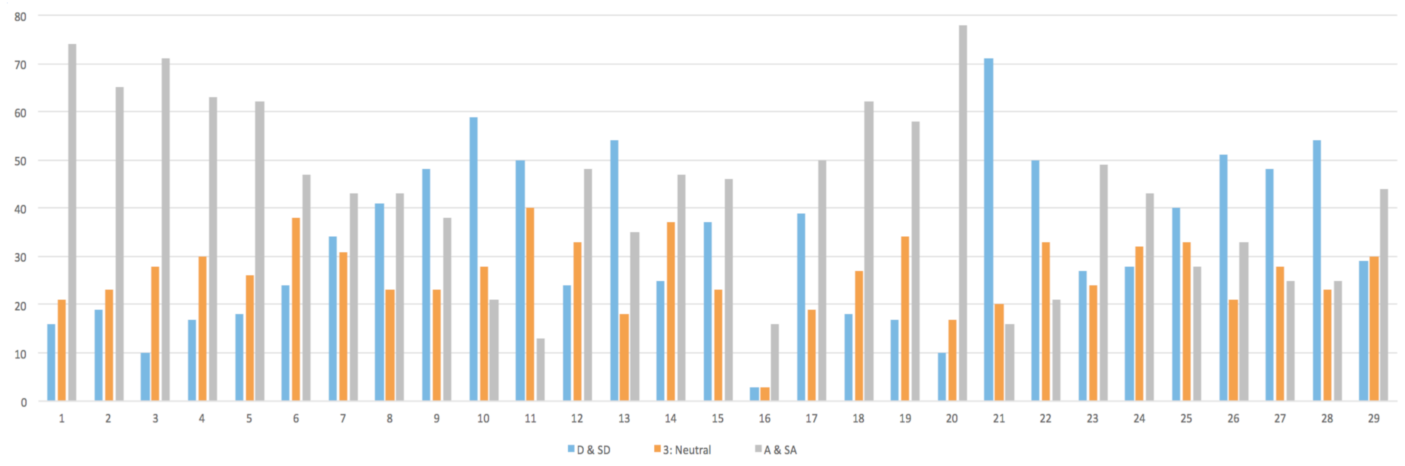

Fig. 1. BGU Results

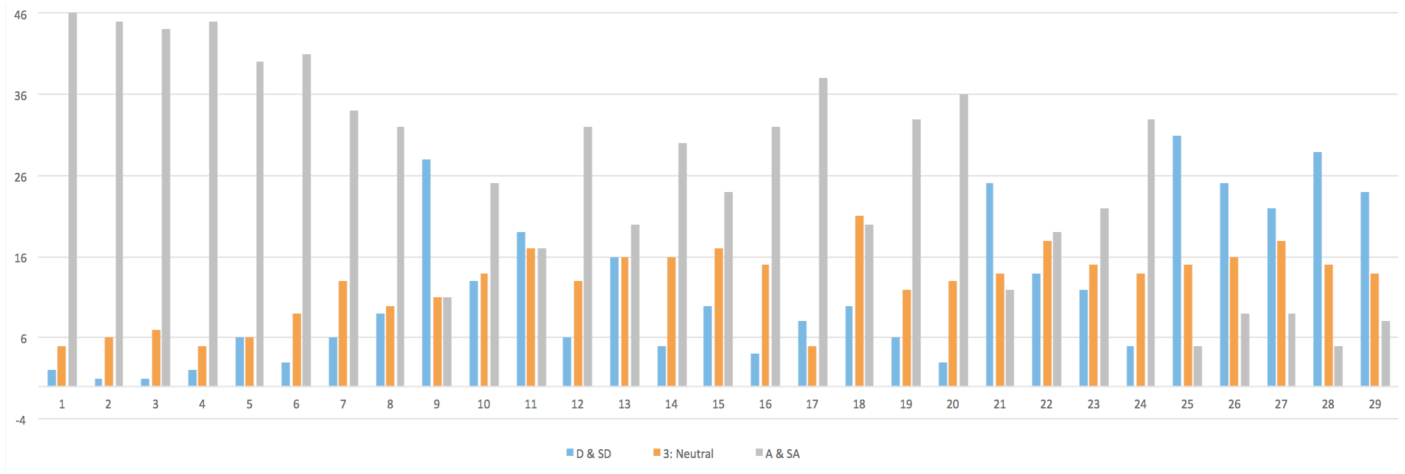

Fig. 2. NAU Results

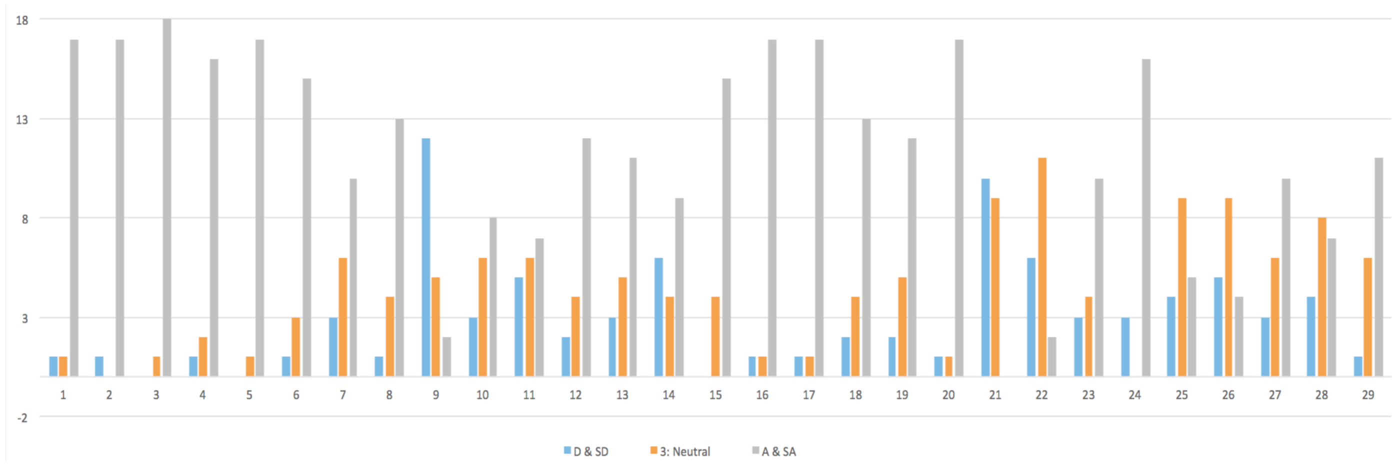

Fig. 3. Concordia University Results

VI. Cross-University Analysis

A. Perception Trends

Of particular interest to this study, is to investigate whether curriculums have positive, negative, or neutral effects on how students perceive the value of modeling. We studied the perception trend of both undergraduates and graduates as follows. For undergraduates, we analyze the changes in perception from year to year, starting from year 1 to 4. For example, if students' average perception of "Models Usefulness" in year 1 is 3.0/5.0 and in year 2 the average perception is 4.0/5.0, this implies that the perception has improved from year 1 to 2. For graduates, we analyze the differences in perception from undergraduates and graduate averages. We do this by taking the average of the entire data set for undergrads and subtract it from the average for graduates.

This analysis is performed using only a subset of questions that reflect models usefulness and value. The subset includes the following questions as listed in Section III.D: 1.a through 1.h, 2.c through 2.g, 2.j, 2.k. 2.l. We also report on the analysis of the students' perception of educating students on modeling using analysis of answers to questions 4.a through 4.d.

We use the sign analysis technique as reported in [18]. We count the number of positives (indicating perception improvement) and negative (indicating perception decline). The results are summarized in the following table.

Table 7. Sign analysis of the survey results

	Usefulness		**Education**	
	+	-	+	-
NAU UG	0	14	0	5
NAU G	5	13	3	2
BGU UG	6	10	0	4
BGU G	3	12	1	3

NAU Undergraduate students' perception of modeling declines as they progress in their undergraduate education, evident by 14 negatives, and 0 positives. This result is also reflected in students' perception of modeling education (0 positive, and 5 negatives). For NAU graduate students, the data is more balanced, but remains overall negative. One possible explanation might be that students with low perception of modeling do not enroll in graduate studies, or that perception of modeling is an indicator of academic success.

The results for BGU are more balanced, but remain overall negative. The perception of usefulness and education among undergraduates and graduates tend to decline over the years of their education.

B. Cross-University Trends

For the cross-university analysis, we are interested in answering the following high-level questions.

Q1. Do students perceive models to be more useful for documentation and communication, as opposed to software development activities (code generation, implementation and maintenance)?

Q2. Do students in general think that more modeling need to be integrated into the curriculum?

Q3. How do students perceive modeling tools? For this question, we limit our data analysis for graduate students as undergrads may not have the sufficient maturity to understand the distinction between the tools, and the approach.

Q4. How does the students' perception change from their undergrad education to their final years in graduate studies?

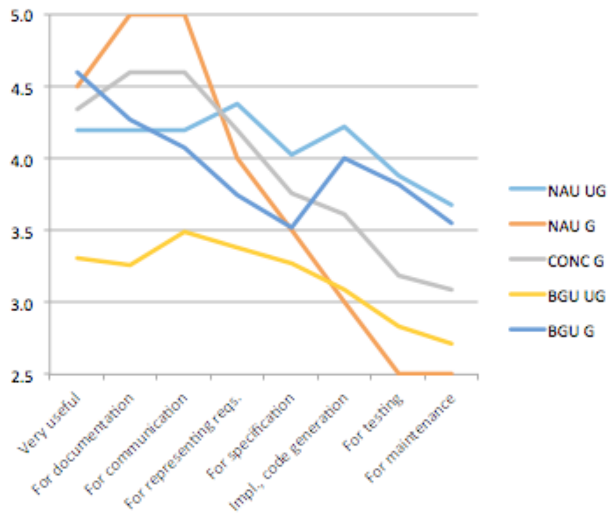

Fig. 4. Models for documentation versus implementation

For Q1, it is evident that students tend to find models more useful for documentation and communication, and less for other development tasks (*Fig. 4*). This preference in perception was more evident in graduate students that undergraduate students, particularly for NAU and Concordia graduate students.

For Q2, both graduates and undergraduates agree that more should be taught about modeling (UG: 3.7, G: 4.0) and that modeling should be taught at the same time (UG: 3.2, G: 3.3), and that modeling should be integrated in computer science and software engineering courses (UG: 3.8, G: 4.0).

For Q3, students did not find modeling tools to be more complex, but rather, modeling tools were perceived to be more difficult to obtain (*Fig. 5*). This is potentially due to the fact that students are introduced to coding much earlier than modeling. They become more quickly familiar with the coding platforms than with the modeling tools.

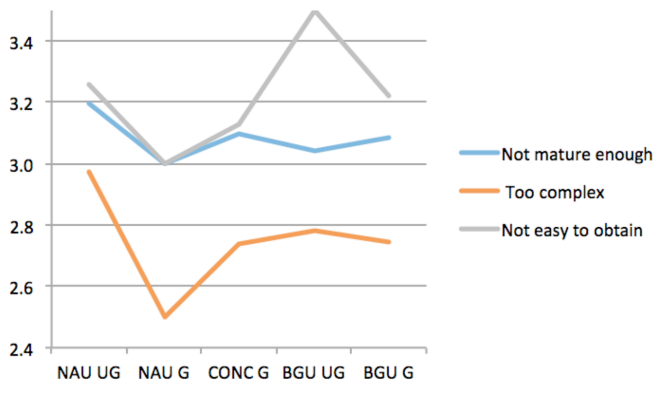

Fig. 5. Perceptions of Modeling Tools

For Q4, we found a consistent pattern of declining perceptions emerging in both NAU and BGU undergrad and grad students. In general, the perception declination was more prominent in the case of undergraduate than graduate students.

This can be interpreted in a number of ways, 1) the curriculum fails to highlight the value of modeling in software engineering or 2) students come to the program assuming unrealistically high value of modeling. During their education years, the curriculum does not improve on that initial perception. 3) For large software projects, students fail to discover the value of modeling (usually, UML), and may be relying exclusively on code. As a result, students may come to the conclusion that modeling is not as useful as they may have thought initially. The lacks of tools may also contribute to this. As the advance in the programs, students usually want to build interesting systems that run quickly so they can make modifications and improve their functionality. The unavailability of good tools make this difficult. This may be a factor that discourages students from adopting the modeling paradigm as advanced stages. This question may require further investigation to uncover exactly why the perception declines.

However, grads tend to perceive modeling more favorably than undergrads, especially for communication, documentation, and tool availability and readiness. This is with the exception of a few aspects of models suitability for software development and testing. This may be interpreted by the nature of the work performed by the grads vs. undergrads. Grads may be using models to abstract ideas and communicate early concepts. Models for such tasks may be more suitable than code.

VII. Threats to Validity

Threats to validity of this study are discussed in this section.

A. Question Bias

The majority of the questions in this study were presented in the positive sense (i.e. models are useful). It is possible that negative questions may have a different impact on how participants respond to questions.

B. Profile of the Respondents

The researchers in this study did not have control on selecting participants. It is possible those participants who opted to complete the survey, or those who decided to complete the survey after it had started, may have had different views on modeling than the general population. Our study collected profiling data and as discussed in this paper, we attempted to analyze the paper while considering the collected profiling data.

C. Different Modeling Teaching Approaches

The three participating institutions deployed different curriculum and different teaching styles. It is possible that the participating universities teaching of modeling may have influenced the views of the participants. This could in effect mean that the participating universities are not a good representation of the general population. This external validity threat was minimized by the fact that three different institutions participated, and that participants were not selected from a specific group or study year.

VIII. Conclusion

This paper reports on a survey that was conducted in three independent institutions. The goal of the survey is to uncover students' perception of the value of modeling in software development. Participants included students from undergraduate year one to last year in the graduate program.

The results suggest that students' perception of the value of modeling declines as they progress in their education. This was true for both undergraduate and graduate students. These results warrant further investigation into why this is the case. The reasons of the decline in perception may be attributed to wrong perceptions of modeling for young students, in adequate coverage of modeling topics, lacks of adequate modeling tools, or immaturity or unsuitability of the modeling techniques and approaches for students' projects. It is also possible that UML is simply not appropriate for defining and/or implementing the problems and solutions they face.

The results suggest that graduate students on average appreciate modeling more than undergraduates. This could be attributed to the nature of tasks that graduate students perform that may be more suitable for modeling approaches.

The results of the study calls for further investigation of the reasons of the relatively low perceptions of modeling usefulness by students. This can be done by further correlations analysis and interviewing students about their perceptions and the reason for that. Furthermore, the results call for revisiting modeling curriculum in order to introduce improvements and to further recruit the students into the modeling era.

References

[1] Omar Badreddin, Andrew Forward, and Timothy C. Lethbridge. "Model oriented programming: an empirical study of comprehension." Procs of the 2012 Conference of the Center for Advanced Studies on Collaborative Research. IBM Corp., 2012.

[2] Andrew Forward, Timothy C. Lethbridge, and Omar Badreddin. "Problems and opportunities for model-centric vs code-centric development: A survey of software professionals." 5th Workshop" From code centric to model centric: Evaluating the effectiveness of MDD (C2M: EEMDD)", University Pierre & Marie Curie, Paris. 2010.

[3] Robert France, and Bernhard Rumpe. "Model-driven development of complex software: A research roadmap." 2007 Future of Software Engineering. IEEE Computer Society, 2007.

[4] Djedjiga Mouheb, Mourad Debbabi, Makan Pourzandi, Lingyu Wang, Mariam Nouh, Raha Ziarati, Dima Alhadidi, Chamseddine Talhi, Vitor Lima. "Unified Modeling Language." Aspect-Oriented Security Hardening of UML Design Models. Springer International Publishing, 2015. 11-22.

[5] Omar Badreddin, Timothy C. Lethbridge, and Maged Elassar. "Modeling Practices in Open Source Software." Open Source Software: Quality Verification. Springer Berlin Heidelberg, 2013. 127-139.

[6] Alan Zeichick, "UML adoption making strong progress." Software development times 15 (2004).

[7] Ariadi Nugroho , and Michel RV Chaudron. "A survey into the rigor of UML use and its perceived impact on quality and productivity." Procs of the 2nd ACM-IEEE int'l sym. on Empirical software eng. and measurement. ACM, 2008.

[8] George G. Mitchell, and James Declan Delaney. "An assessment strategy to determine learning outcomes in a software engineering problem-based learning course." International J. of Engineering Education 20.3 (2004): 494-502.

[9] Chanan Glezer, Mark Last, Efrat Nachmany, Peretz Shoval. "Quality and comprehension of UML interaction diagrams-an experimental comparison." Information and Software Technology 47.10 (2005): 675-692.

[10] John Hutchinson, Jon Whittle, Mark Rouncefield, and Steinar Kristoffersen. Empirical assessment of MDE in industry. In *Proceedings of the 33rd International Conference on Software Engineering* (ICSE '11). ACM, New York, NY, USA, 2011. 471-480

[11] Timothy C. Lethbridge, Gunter Mussbacher, Andrew Forward, and Omar Badreddin . "Teaching UML using umple: Applying model-oriented programming in the classroom." Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on. IEEE, 2011.

[12] Dirk Frosch-Wilke, "Using UML in software requirements analysis-Experiences from practical student project work." InSITE-Informing Science and IT Education Conference. 2003.

[13] Martin Grossman, Jay E. Aronson, and Richard V. McCarthy. "Does UML make the grade? Insights from the software development community." Information and Software Technology 47.6 (2005): 383-397.

[14] Richard M. Felder, and Rebecca Brent. "Designing and teaching courses to satisfy the ABET engineering criteria." J of Eng. Education-Washington- 92.1 (2003): 7-26.

[15] João Paulo Almeida, Remco Dijkman, Marten Van Sinderen, and Luís Ferreira Pires. "Platform-independent modelling in MDA: supporting abstract platforms." Model Driven Architecture. Springer Berlin Heidelberg, 2005. 174-188.

[16] Marian, Petre. "UML in practice". In *Proceedings of the International Conference on Software Engineering* (ICSE '13), 2013. IEEE Press, Piscataway, NJ, USA, 722-731.

[17] First International Workshop on Human Factors in Modeling (HuFaMo'15). Available: http://hufamo.compute.dtu.dk/.

[18] W. J. Dixon and A. M. Mood. "The statistical sign test". 1946.
Journal of the American Statistical Association pp. 557-566.