Students' metacognitive processes and impact on Self-efficacy in embedded programming

Ole Schultz, Department of Engineering Technology and Didactics, DTU
Denmark
osch@dtu.dk
Tomasz Blaszczyk, Department of Engineering Technology and Didactics, DTU
Denmark
tomb@dtu.dk

ABSTRACT
Keywords - Metacognitive process, self-efficacy, Emotion, Vignette questions

For minimizing students drop out on 2nd semester, Electrical Engineering (EE) BEng we experiment with a written and video process guideline for support of solving programming problems and metacognitive awareness. We will try to measure how students emotional experience programming by using a special self-assessment vignette inquiry. On 1st semester, we will measure when programming as novices in two study lines (EE - and IT-Electronic BEng students (IE)) and do a comparison with 2nd semester for EE students. On 2nd we introduce a process for program development in Digital electronics and programming (DEP) and we will measure 3 times during the semester the effect of the process by using self-assessment vignette inquiry. The working hypothesis is: Can the emotional experiences become lower, then the self-efficacy will be higher and the drop out will be lower. The articles describes the theoretical background for both the process and the students' self-assessment resulting in emotional experiences. The results so far are that on 1st semester IE there is only 20% of students, which has a total score greater that 40 (total score max 78) whereas among the 2nd semester EE students 33% students has a score above 40. High score means great emotional impact.

I INTRODUCTION

This article here is part of a project running in DTU Scholarship of Teaching, where we wonder about that through several years, we have experienced that few students are dropping out from taking the exam in programming courses in the first two semesters at Electrical Engineering (EE) BEng programme and IT Electronics (IE) BEng programme. During the past years, we have observed that more students have difficulty to figure out how to proceed and cope with a so-called compiler message, or when the program does not work as expected. They do not understand what to do in the process of programming. After conducting several interviews, we identified that students drop out or do not take the exam due to their programming difficulties and low self-efficacy. On 2nd semester in DEP, 5-10% of the students who persist in the first part of the semester express that they do not how to start the programming an assignment and find it exceedingly difficult understanding how to use binary operators in C-programming.

Research question

Our hypothesis: If students get a process for tackling problem presented in the course, then they will get more self-efficacy and thereby the motivation for learning should increase. That leads to the question: 'Can metacognitive processes help students to gain more self-confidence and thus continue to be active during the course?'

Blended learning used in the DEP

We use the approach of so-called blended learning as teaching method, which requires that students prepare before attending traditional face-to-face lectures. For comparison and for future improvement we studied (Alammary, 2019), where he did a systematic review on blended learnings models used for introductory programming courses. The course content is a mix between understanding the hardware/digital electronics and programming registers in microcontrollers. Assignments are about communicating data to and from the microcontroller and operate on the data.
Thus, pedagogical method is blended learning, and with a reference to (Alammary, 2019), the method is called "Supplemental model", which means that online activities is added to the course and connected to activities in the class. The online activities before each lecture are video recordings presenting digital electronics and programming tutorials, online conceptual and programming quizzes. Typically, during the face-to-face lecture, the class starts by reviewing answers to questionnaires and discussing the results. This followed by a presentation with introduction to relevant parts of the theory (for example about the microcontroller or the C-construction) needed for solving the assignments. There are five assignments for hand-in during the 13 weeks course, where four of these includes an assignment report. The students work in groups of 2-3 students. They have three hours for solving the assignments, with supervision by lecturer and teaching assistant.

II METHOLOGY

For answering the question, we have studied some articles dealing with how to teach in programming and how students' self-asses their ability and how the process of programming has an impact on the self-assessment.

Literature studies - related work

When we use Self-efficacy as a term, we found in (Bandura A. 1977) his definition we find useful. Self-efficacy perception understood as "beliefs in one's capabilities to organize and execute the courses of action required to produce given attainments" Bandura A. 1977.

For answering the hypothesis and research question, we have done several studies regarding criteria students use to evaluate their programming ability. For example, (Lewis, C. M. et al. 2011) mention students think about speed and grades. In (Gorson, J. et al. 2020) and their prior work, students' thoughts about looking up syntax and getting errors are signs of low ability. Gorson found some of the criteria contradicts with what instructor's think are important for novice programmers' success or professional practices.
The authors suggested that students', to their opinion, inaccurate expectations of the programming process could have an impact on how they self-assess. (Kinnunen, P. et al. 2012) have studied how the students' emotional experiences during the programming process relates to the self-efficacy assessment. They found the programming process has an impact on the students' experience with self-efficacy and their expectation. Criteria such as fluency, and time spent on assignment has an impact on their assessment of their abilities. They also compare themselves with other students and how those progress in solving assignment and the time spent. For instance, students are feeling bad, because other students managed to finish faster. External factors as working together can also have a negative or positive impact on the self-assessment, where supporting partner relationships, partners helping each other contributes to positive experiences. Whereas in the case of unsupportive partner relationships, the partners direct negative feedback directly contributed to negative self-efficacy. This work does not considered the groups' relationships factors.
The assignment formulation can have an impact on student's self-efficacy. For example, is it not understandable, or if it is not obvious what to do, it can result in a negative self-assessment of the

abilities. In contrast to this, in literature study we found, the students do not believe that the teacher will give an assignment that cannot be solved, so even if it hard to understand this can make a positive impact on the self-assessment.
In (Gorson, J. et al. 2019) they discuss the students' mindset and it's influence on the students perceived ability and persistence in Computer science. We also think it has an influence on the perceived self-efficacy. Gorson pointed out that research in psychology has demonstrated that students' beliefs about the malleability of intelligence can have a strong impact on other reactions to challenge and academic performance. Literature (Dweck, C. S. 2006, Loksa, D. et al. 2016, Prather, J. et al. 2019) concludes that the mindset theory about Growing mindset and Fixed mindset have an influence on the learning and approach to problem solving. Students with Growing mindset are more likely to persist challenges.

Programming Process guide

In (Loksa, D. et al. 2016) they describe and discuss problem solving stages and metacognitive prompts. They propose two interventions that teach learners how to converge toward programming solutions while teaching them how to recognize, evaluate and refine their problem-solving strategies. One is to provide students with explicit instructions on the goals and activities involved in programming problem solving, while another is about using an explicit questions technique. When students want advice, they were asked about where in the programming process they are.
A study by (Prather et al. 2019) did an experiment for investigating whether an explicit metacognitive prompt discussion and if a process guide support metacognitive awareness. In (Falkner, K. et al. 2014) they discuss how they can assist students in self-regulated learning strategy. The study proposes an example guide to development of scaffolding activities to assist learning development. (Falkner, K. et al. 2014) propose introduction of diagrams class diagrams or flow charts, assessment of the task difficulty, identifying the needed skills - leading to time management and sub goal plan. Therefore, it is important to conceptualize the design by diagrams as a part of the software development process, and link it to the planning tasks. At the same time the conceptualizations means, it can change during the programming process and therefore viewing it as an iterative approach. It can help explicit inclusion of experimentation as a part of the design, exploring alternative design, evaluation, and comparisons. Both studies have inspired us to formulate the process guidelines shown here below. We adjusted and added further questions to be used in the Digital Electronics and Programming ourse (DEP).

Process guide

In the first lecture in the 2nd semester EE class in Digital Electronics and Programming, we introduce a process guide sheet to support the process of programming. We want to measure the effect of using self-assessment vignette questionnaires in 1st week, the 6th week and the 12th week, for measuring the experiences of programming when students use the process.

The process guide:

1. Read the whole assignment. Does the assignment make sense?

2. What could a solution to the task /subtasks look like? Outline the solution with a pseudocode and / or a flow-chart.

3. Imagine a simulated execution of your hypothetical program / parts of the program. Use your pseudocode and flowchart. Simulate that you provide running the hypothesis program. Does the expected happen?

4. What can the C-code for the sub-task / task solution look like?

5. Open Microchip studio, start a new project, select GCC C executable, give the project a telling name and place the project somewhere where you can find it again. press ok. Choose to use an ATmega2560. Program the solution you have found for each sub-task found under 2.

6. Does the program perform the hypothetical run-through performed in. 3?

The link to the full process guideline is in the reference (Schultz, O., 2021)

III DATACOLLECTION

Measuring impacts from programming situations

(Gorson, J. et al. 2020) compared three different universities undergraduate students in Computer Science. In his study, he shows students who self-assessment negative tend to have lower self-efficacy and concludes "We also found that the frequency with which students negatively self-assess correlates with their overall self-efficacy in their programming courses". For data collection, Gorson uses a summative survey together with interviews. The survey is interesting as he uses a vignette survey, for measuring 13 distinct moments in programming and how they influences the feelings.
We find this method interesting to use in our study especially measuring if the process with guideline has a positive impact on the student self-assessment of problem solving while programming. The vignette questions used by (Gorson, J. et al.) 2020 relates to the professional way of working with programming, therefore it is relevant to use for our BEng students. As they are educated to conduct the professional practice of engineering.
We used the 13 vignette questions shown in (Gorson, J. et al. 2020), but have translated them to Danish and we have substituted the person names with 1st and 2nd person singular subject pronoun. The reason for not using original questions is that students find it hard to follow another person's feeling - therefore we adjusted appropriately. We also used another scale, from 1 to 6, where 6 is most negative and 1 is least negative. The vignette statements presented in Table 1.

Table 1 Vignette statements

A. A Simple Mistake:	You are working on your programming task. You compile its code. An error is displayed. You immediately realize that she has omitted a parenthesis. You add parentheses. How does it affect you?
B. Start over:	You are working on a hard programming task. You are planning a solution. You write a few lines of code. You realize the approach to the problem is not working. You decide to start over. How does it affect you?
C. Do not understand error message:	You are working on a programming problem. You compile your code. An error is displayed. You have no idea what the error message means. You are not sure what to try to do. How much does it affect you?
D. Stop programming to plan:	You start working on a programming problem. You write a few lines of code. You realize I am confused about what the next step is. You pause and plan your next step. You wish you did not have to stop writing code to plan. How much does it affect you?
E. Get help from others:	You are working on your programming task. You are stuck. You get help to complete the task from the teacher or assistant teacher. How much are you affected?

F. Spending time looking up syntax:	You are working on a programming problem. You cannot remember the syntax. You use Google to look up syntax. You are disappointed that you could not remember the syntax yourself. How much are you affected?
G. Spending time planning in the beginning:	You are unsure how to start your programming task. You spend time planning how to solve the problem. Eventually, you come up with a plan and start writing code. You wish you did not have to spend so much time planning before writing code. How much does it affect you?
H. Spends a lot of time solving a problem:	You work hard at programming to solve a programming problem. You solve the problem. You are proud of yourself. You look at the clock and realize how many hours you spent on the problem. You feel sorry for it because it took so long to fix it. How much does it affect you? How much does it affect you?
I. Do not know how to get started:	You solve your programming task. You open the program editor but have no idea where to start. You feel disappointed in yourself because you do not even know how to get started. How much does it affect you?
J. Spends a long time finding a simple mistake:	You are working on a challenging problem. You are running into a mistake. You are looking through the code but cannot find the error. After a long time, you realize that it was a small typo. You think to yourself, "Wow. I am so bad at programming. A good programmer would not take that long to find a simple error." How much does it affect you?
K. Struggling to find the error:	You are working on your programming homework. You run your program and get an error. You struggle to correct the mistake for a long time. After the error correction, the program runs, and another error occurs. You fight again. Eventually you solve it. How much does it affect you?
L. Unable to complete within expected time:	You are working on your programming task. You expect to finish it in one night. After a while, you decide to stop work because it was late. You feel sorry that she was not able to finish it in one night. How much does it affect you?
M. Do not understand the problem of the task:	You solve your programming task. You do not understand what the task asks you to do. You feel sad and frustrated because she cannot even understand the question. How much does it affect you?

The vignette in Danish is in the references.

The sentences to the left cover moments in programming process and to the right there are related sentences about what thoughts are coming up. In another study (Kinnunen, P. et al. 2012), described six stages of experiencing programming: getting started, encountering difficulties, dealing with difficulties, succeeding, submitting, and stopping. Stopping can happen without submitting because of struggling with difficulties. In table 2 we have mapped the 13 vignette statements to the six stages of experiencing programming, finding what type the vignette statements evaluate in the six states of experiencing programming. As can be seen in table 2, the vignette statement will measure mostly on how to handle difficulties. It does not measure the release of stress by submitting. The reason for that is the 13 moments of programming is while we are in the process of developing a program.

Table 2 Six stages of experiencing programming

Stage	vignette questions
Getting started	B, D, G,L,M
Encountering difficulties	A,I,J,L
dealing with difficulties	C, E, H, J, K, L
succeeding	H
submitting	
stopping	C, D,I

IV RESULTS AND DISCUSSION

Vignette answers

We collect answers during the first 4 – 5 weeks in September 2021 from three different classes two on first semester and one on 2nd semester. Before we used the vignette enquiry, we did a test in June in 2nd semester class at EE study line and these results are presented here as well.
After 3 weeks we asked It-electronics - IE students on 1st semester in the course Introduction to Embedded Systems and the results are shown in figure 1. In figure 1 and figure 2 we have all answers for each student summed up, on the y-axis is the sum of scores of each vignette, on the x-axis is p1 to pn each students answers. Vignette A has the most dark grey colour and lightest for vignette M. The course number is included in the headline.
Referring to figure 1 and table 1, in figure 1 to the left 26 students answered out of 37. We assume it is the active students who responded. 20% of the IE students has a total score above 40. In the figure 1 to the right, we have unfortunately only 18 responses out of 55 1st semester students enrolled in the course 02318 at EE. If the results is regarded representative then the impact is much higher that 1st sem. IE. As 55% has a total score above 40. This fact can be explained by that IE students received very strong motivation in first few lessons by demonstration of previous results achieved after 1 semester by engaged students. Whereas 1st semester EE students did not receive any demonstration

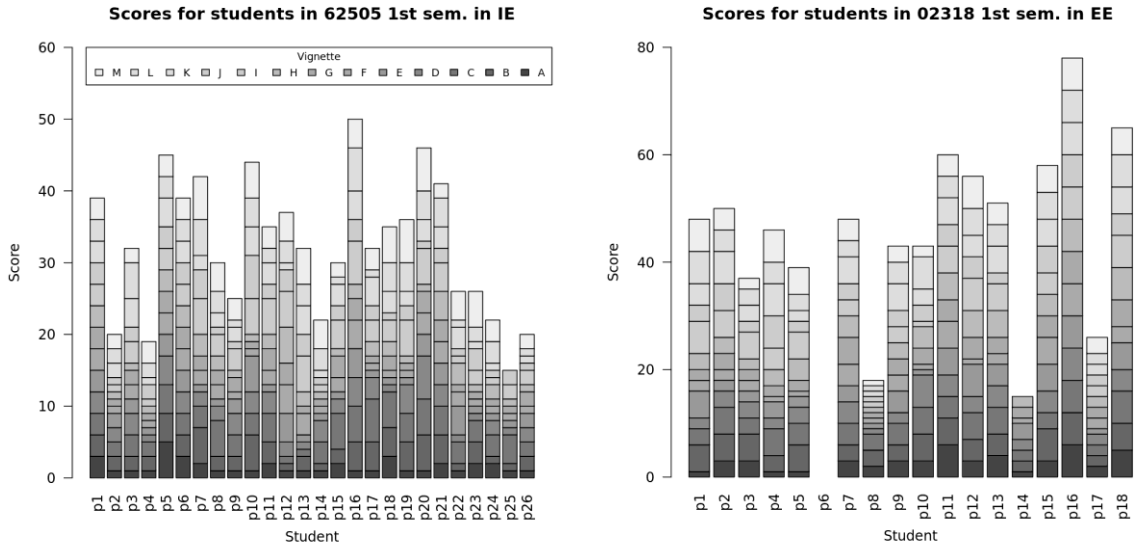

Figure 1 Sum score for: 1st sem. IE students (26) and EE 1st sem. Students (18)

We also asked the students if they had been programming before enrolment on the study and 85% had

programmed before in the IE class. Whereas in the EE 1st sem. class 50% out of the 18.
That can be another reason for the difference between the two first semester classes. Another reason can be the IE students have three different course dealing with programming on 1st semester, whereas the EE students only have one course. For now, we do not know if there is a correlation between the lower score and the previous experiences, but it has to be further evaluated.
In figure 1 to the right, P6 in the EE 1st semester did not answer the vignette statements therefore empty and the p16 has just scored six for all vignettes, which perhaps is unserious. The authors does not have the class and the students got an e-mail with the link in and this had been repeated three times each time with explanation about why they should answer.
Figure 2 shows the sum of scores for 2nd Sem. EE students (27) out of 33 and 2nd sem. EE students (20) out of 45 students. Students here have had the 1st semester programming course and when having the 30082 course they have had the 2nd semester DEP (62734) course.

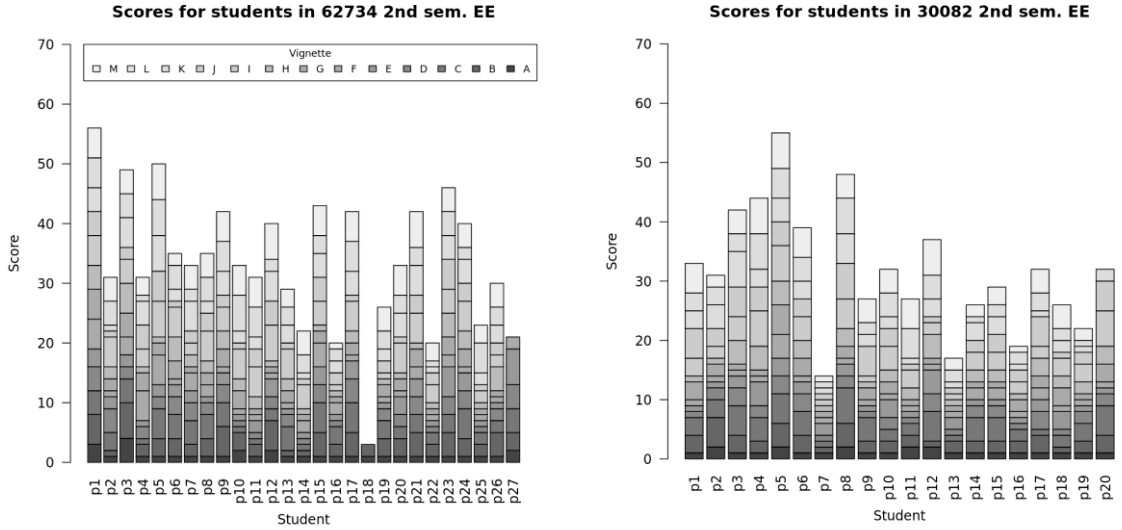

Figure 2 Sum score for: 2nd sem. EE students (27) and EE 1nd sem. Students (20)

In the first lesson on 2nd semester in DEP (62734) we asked the students to do the vignette inquiry before introducing the course. 33% of the responses show a score above 40 – where maximum is 78 – Compared to the IE students they are more influenced, it could be lack of previous programming experiences, which can explain it – That must be evaluated further.
In June as a pre-test, we used vignette questionnaire on a 2nd semester class running in 3 weeks – 20 students answered (45% of the whole class) in digital design (30082) where they are using the 13 weeks course DEP course together with another 13 weeks course Digital design(30082). Results presented in figure 2 to the right.
It seems only 15% total score is above 40 in the right figure, which is lower that the results in figure 2 to the left – it could be the representation of students(only 20) is not representative or due to their programming experiences they do not become so hard influenced.

For overall comparison in the figure 3 (next page) a boxplot is chosen to reveal differences between the 13-vignette statements for the four classes. Y-axis shows the possible score for each vignette. The X-axis show the individual Vignette statement by Letter A to M from Table 1. The vertical rectangular box horizontal line shows seen from top respectively the upper 3rd quartile and the lower 1st quartile. The lines going up and down from the box is showing the spread in the dataset up to maximum score and least score. The bold line between the two boxes indicates the median value. Open dots shows Outliers. We will here take a closer look in some the values found in figure 3. When comparing the 1st semester's responses, it is clear the 1st semester students at EE have a general higher median value and 3rd quartile for the most of the vignettes compared to the 1st semester students from IE.

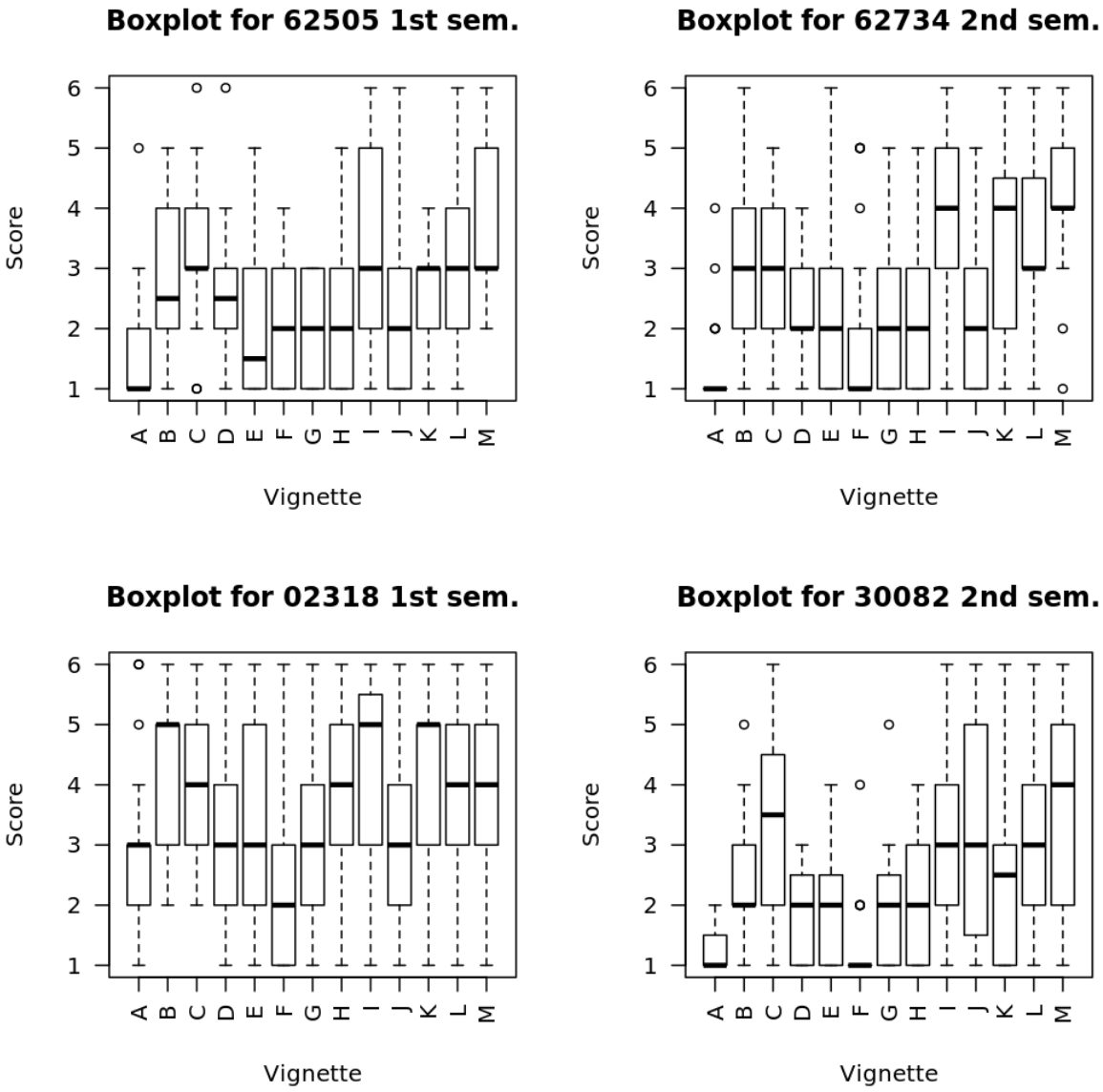

Figure 3 Boxplot for the students' answers

The vignette statement K "Struggling to find the error" has a median score of 5 and for the 1st semester students EE and for IE it is 3. For all four classes of students the M "Do not understand the problem of the task" has the high negative impact the 3rd quartile is 5 and the median value is 4 except 1st semester students at IE study.
The box plot to lower left corner in figure 3 shows students at 2nd semester 3-week 30082 course. These students have answered while they were terribly busy solving their project and only 40% answer – We use the data but is aware about it is only 40% who has answered.
For 1st and 2nd semester EE students the I, K L, M vignette is showing most negative impact -the I: "I do not know how to start", the K: "Struggling to find the error" L: "Unable to complete within expected time" and M: "Do not understand the problem of the task". In Table 3 is listed the most significant results for the 4 vignette (I to M) in numbers of students and the boxplot median values Where the median values is respectively found in figure 3, from left upper corner to lower right corner in figure 2

Table 3: Results for vignette I to M.

Vignette	Stud. course	Median	Students Score >=4
I	1st EE 02318	5	11 61%
	1st IE 62505	3	8 31%
	2nd EE 62734	4	18 67%
	2nd EE 30082	3	7 35%
K	1st EE 02318	5	11 61%
	1st IE 62505	3	4 15%
	2nd EE 62734	4	15 56%
	2nd EE 30082	2.5	<4 <12%
L	1st EE 02318	4	11 61%
	1st IE 62505	3	8 31%
	2nd EE 62734	3	13 48%
	2nd EE 30082	3	7 35%
M	1st EE 02318	4	11 61%
	1st IE 62505	3	12 46%
	2nd EE 62734	4	13 65%
	2nd EE 30082	4	21 77%

From table 3 we conclude that the M (Do not understand the problem of the task) has the absolute strongest negative impact on the students' feelings.

Aspect of the programming is error messages and finding faults. The Vignette J: "Spends a long time finding a simple mistake" and C: "Do not understand error message" show the impact.
Table 4 lists the results

Table 4: Results for vignette C and J.

Vignette	Stud. course	Median	Students score >=4
C	1st EE 02318	4	11 61%
	1st IE 62505	3	11 42%
	2nd EE 62734	3	12 44%
	2nd EE 30082	3.5	10 50%
J	1st EE 02318	3	7 38%
	1st IE 62505	2	2 <10%
	2nd EE 62734	2	5 19%
	2nd EE 30082	3	8 40%

It most negative impact has the understanding of error messages, which means the compiler responses, can be a challenge to understand and that is for all 4 classes. The influence of spending time is most influencing the EE classes
When using the table 2 above as a classification with the 6 stages of experiencing programming, then the vignette answers is in the two difficulties stages "Encountering difficulties" and "dealing with difficulties" is the I and M part of.

Discussions
When the results above is interesting compared to the process guideline. As the process, guideline is as the first question is Reading the assignment text and "Does it make sense?" and step 2. If understanding and finding part problem is clear, it should minimize the negative influence from M and gain the self-efficacy. If it is clear what each part of the program should do then the struggling finding errors could be minimized. But if the error is about understanding the compiler message, then it is experiences which will do I, which perhaps is the reason for 1st semester IE students' responses has low score- due to privies experiences with programming.

Introduction of the process on 2nd semester EE

In the second lesson, the student should in groups of 2 to 3 student start doing the exercise 2. I orally introduced the process guideline. The process description were before the lesson uploaded to Learn (CMS system). I told them to use the process described and make ex. make flow chart before coding. Observation where that only very few students did do process work, they open the programming IDE and started up beginning writing code and discuss which register, and which bits should be set. As supposed in the literature, the slides for the lesson2 introduced the code snippets- so they could get some hints – regarding the step 4 in the process description sheet. Even the exercise text was adjusted compared to earlier semesters text, so it was pointed how the process step should be done, they just started writing code.
In week 4 they got the next exercise – and there I intervened – so they were all asked to go through the process, understand the text, find sub-problems sketched flow charts for each part. They got 30 min. And then we discussed their findings and, on the whiteboard, sketch some flow charts – I asked how they find that, and the answers were that it had been fine – I seems helping them that I wanted them to deliver on the process.

V CONCLUSSION AND FUTHER WORK

In this paper, we collected results from 71students from 1st sem. IT-Electronic, EE, and 2nd sem. EE. Interestingly, we noticed significant difference in questionnaire answers between two study lines at the 1st semester. One questions rises here does the students background, before enrolment have an influence on their answers. We found if the students don not understand what the task is about can have a negative impact on their self-assessment. Therefore, the process should help lowering it. In addition, we had become more aware about how important the text description of assignments are. We still have to collect two times vignette inquiries from the 2nd semester class. These results is important for answering the hypothesis. At the conference, further results will be shown and discussed. At the conference we should be able to present if the process lower unregistering from exam.
We think that future work might focus on investigation how to measure and where we measure and to what extend the process has a motivation factor and can influence self-efficacy then other factors such as social networking can be performed.

ACKNOWLEGDEMENTS

We would like to acknowledge special consultant Maria Svendsmark Hansen, DTU engineering technology for giving feedback on this article and associate professor Hanne Løje for making it possible to take part in Scholarship of teaching work with the possibility to write this article.

REFERENCES

Alammary, A. (2019). Blended learning models for introductory programming courses: A systematic review. *PLOS ONE*, *14*(9), e0221765. https://doi.org/10.1371/journal.pone.0221765
Bandura, A. (1977). *Social learning theory*. Pearson College Division.

Dweck, C. S. (2006). *Mindset: The new psychology of success*. Random House.

Dweck, C. S. (2013). *Self-theories: Their role in motivation, personality, and development*.

Psychology Press.

Dweck, C. S., & Leggett, E. L. (1988). A social cognitive approach to motivation and personality.

Psychological Review, *95*(2), 256–273. https://doi.org/10.1037/0033-295x.95.2.256

Falkner, K., Vivian, R., & Falkner, N. J. G. (2014). Identifying computer science self-regulated learning strategies. *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education - ITiCSE '14*. http://dx.doi.org/10.1145/2591708.2591715

Gorson, J., & O'Rourke, E. (2020, August). Why do CS1 Students Think They're Bad at Programming? *Proceedings of the 2020 ACM Conference on International Computing Education Research*. http://dx.doi.org/10.1145/3372782.3406273

Gorson, J., & O'Rourke, E. (2019, July 30). How do students talk about intelligence? *Proceedings of the 2019 ACM Conference on International Computing Education Research*. http://dx.doi.org/10.1145/3291279.3339413

Kinnunen, P., & Simon, B. (2012a). My program is ok – am I? Computing freshmen's experiences of doing programming assignments. *Computer Science Education*, *22*(1), 1–28. https://doi.org/10.1080/08993408.2012.655091

Lewis, C. M., Yasuhara, K., & Anderson, R. E. (2011, August 8). Deciding to major in computer science. *Proceedings of the Seventh International Workshop on Computing Education Research*. http://dx.doi.org/10.1145/2016911.2016915

Loksa, D., Ko, A. J., Jernigan, W., Oleson, A., Mendez, C. J., & Burnett, M. M. (2016a, May 7). Programming, problem solving, and self-awareness. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. http://dx.doi.org/10.1145/2858036.2858252

Prather, J., Pettit, R., Becker, B. A., Denny, P., Loksa, D., Peters, A., Albrecht, Z., & Masci, K. (2019, February 22). First Things First. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. http://dx.doi.org/10.1145/3287324.3287374

Schultz, O (2021) link to full process guide Link to full text for the process guideline. https://docs.google.com/document/d/1EThYp665TZNaXE12FkFGHiDs90v0drZopjVNfNujQmQ/edit

Schultz. O. (2021) link to the vignette inquiry https://forms.gle/AxYUVXLGENKXGPg98

BIOGRAPHICAL INFORMATION

Ole Schultz, associate professor at DTU Engineering Technology and didactics department of Internet of things and digital security. Giving lectures in embedded programming and internet of things in homes. VHDL in digital design. Running projects in cooperation with the industry. Taking part in cross-disciplinary networks with teaching and learning and UN Sustainable development goals.

Tomasz Blaszczyk: assistant professor at DTU Engineering Technology and didactics department of Internet of things and digital security. Giving lectures in embedded programming and internet of things, security in embedded systems, radio communication with Narrow band IOT and Lora wan. Running projects in cooperation with the industry.