What competence do software companies want from university graduates?

T Stålhane, *NTNU, Norway*

B Deraas, *NTNU, Norway*

G Sindre, *NTNU, Norway*

Abstract

An important concern for study program design is the employability of candidates. This paper seeks to establish what competence is sought by employers of IT candidates, based on interviews with representatives from 120 Norwegian companies working with IT. The results have been analysed to identify what the IT industry expect from university candidates. For the whole sample, the most frequently mentioned characteristics were programming knowledge, adaptability, the willingness to learn new things and problem-solving skills. If we only consider the companies that develop software, the most frequently mentioned characteristics were programming knowledge, the willingness to learn new things and communication skills. The interviewees were also asked which characteristics that they were missing from university candidates. The most mentioned shortcomings were practical experience with coding and maintenance, adaptability and communication skills when relating to clients. Towards the end of the article, it is discussed how the university can improve in these respects.

The industry's focus on adaptability and the willingness to learn new things indicates that the time when it was considered important that graduates knew how to use specific tools is over. The IT-industry has come to realize that they cannot escape a future of frequent changes to tools and methods.

1. Introduction

Universities are expected not only to give candidates academic knowledge, but also to prepare them for work (Fallows and Steven, 2000), and "Working life" is one of eight main categories of questions in the National Student Survey[1] that NOKUT runs annually in Norway. A good score here can be encouraging. As an example, NTNU's five-year integrated master in Computer Science scores 4.2 on the five point Likert scale question "I acquire skills and knowledge that are useful in the labour market", and a low score might indicate challenges for a study program. Still, such surveys provide limited aid for improvement of study programs, since they lack detailed information about strengths and weaknesses in the candidates' competence profile. Hence, universities need to make their own investigations, and preferably not just with students, but also with employers, who have more direct knowledge about the job opportunities for various competence profiles (Markes, 2006).

At the same time, there are good arguments for involving students more in research already at the early undergraduate level (Healey and Jenkins, 2009). Hence, in the course TDT4140 Software Engineering at the NTNU, Spring 2017, it was decided that part of the compulsory project work in the course would have students interview IT industry representatives about what competence a graduate should have to be employed in that company. The interviews partly had a motivational function, helping the students to see the relationship between course content and work-life needs, and only

[1] https://www.nokut.no/en/studiebarometeret/the-national-student-survey/

constituted a smaller "warm-up" part of the compulsory coursework, the main part was about conceiving and developing a prototype for an app. The learning outcomes of that development project has been analysed in another paper (Kolås and Munkvold, 2017). The current paper, on the other hand, does not focus on what the students learnt from the exercise, rather our research questions are as follows:

- Based on the students' interview data, what competencies do industry employers prioritize when hiring IT graduates?
- What are the most notable gaps between the learning outcomes of our IT degrees and industry needs? And how can these gaps be reduced?

The rest of this paper is organized as follows: we give a short summary of some of the related work, followed by a description of the data collection and how we analysed the data we collected. We then describe how we used a set of requirement categories to identify which competences the industry wants and what they are missing based on the data analyses. The paper ends with a discussion of threats to validity, some tentative conclusions and some thoughts about further work.

2. Related work

There are several ways to investigate what competence is needed in various professions. Document studies of job ads for vacant positions is one possibility, some examples for the software development profession are (Surakka, 2005) focusing on technical competence, and (Ahmed et al., 2012) focusing on soft skills. More common, though, is the usage of questionnaires and interviews, with employers and / or alumni. Questionnaire surveys with alumni typically investigate what competencies they have needed at work vs. what they learnt at university. Classical examples are (Lethbridge, 2000, Kitchenham et al., 2005), both finding some subjects to be over-taught vs. industry needs for software engineers (e.g., maths, chemistry, formal methods), and other subjects to be under-taught (some technical ones, plus most notably business and soft skills like management, leadership, negotiation, presentation skills. Alumni surveys with similar questions have also been performed computing masters at the NTNU (2007, 2011, 2015, next one planned for Autumn 2019) for the purpose of study program QA, though not taken to the level of scientific publishing, except for a paper looking at parts of the 2007 and 2011 data to specifically analyse the need for software testing in the curriculum (Deak and Sindre, 2013). A study of alumni from a Philippine university (Aguila et al., 2016) had many similar findings as the previous studies, but also some differences. For instance "Love of God" came up as a highly valued treat of employees, which would less likely have been the case in more secularized countries, illustrating that even if the software industry is highly globalized, regional and cultural differences also come into play in assessing employability.

Most closely related to this paper are other papers reporting on interview studies with employers. Two notable investigations in Norway are (Lauvås and Raaen, 2017, Lundberg et al., 2018). While technical competence was surely valued, a key finding from both studies was that soft skills like communication and organizational understanding were also essential, and to some extent prioritized higher. As concluded by the former paper: "Most importantly, all participants valued non-technical skills highly. Between *enthusiasm* and *curiosity* and the two somewhat overlapping categories *teamwork* and *cultural fit* we find what seemed most important to all companies". (Lauvås and Raaen, 2017, p.10). Studies in other countries have had similar findings. (Finch et al., 2013, Hamilton et al., 2015, Marks and Scholarios, 2008) all found signs of increasing emphasis on soft skills like teamwork and communication. A notable difference between the above-mentioned interview studies and the one reported upon in this paper, is that our study used students as interviewers, and the original purpose was not research for a scientific paper, rather it was an educational and motivational activity to help them see the connection between curriculum content and work-life needs.

3. Data collection process

All the students in the course TDT 4140 (Software Engineering) received the following instructions:

"Perform a small interview [of a person involved in employment decisions in a company working with IT] with the following three questions:

1. What skills are expected (or most valued/needed) from a person working as a software developer in your company?
2. Why these skills are important?
3. Based on your experience, which skill(s) – if any – are typically lacking from students graduating from a University or a College.

Make notes in the interview and as a deliverable write a brief summary of your findings and observations, including a few words about the person you interviewed (position, company and how many years of experience)."

Close to 400 students took the course, working in groups of three on the assignment. Having each group interview a different company yielded data from 120 companies. The sample contained 65 software companies – eight that wanted to be anonymous, 11 hardware companies, four research institutes, two universities, 17 in the category "other" and 21 companies that did not want to be assigned a company category. Of these 21 companies, eight were also unidentified. The company name (if available) and answers to the three questions, were registered in an Excel sheet.

The 13 named companies in the "none" category that were identified by names were mostly consultancy companies (eight). In addition, there were two media companies, two system development companies and a construction company. Companies without category and name were removed from further data analysis.

Not all the datasets were complete – one or more information items are missing in several cases. We thus have to choose between two alternatives. (1) Throw away all incomplete data sets and only use the complete data in the analyses. (2) For each analysis, use all the data sets that had the required information. We ended up choosing alternative (2). The main reason for this is that it is important to have as many data points as possible in order to be able to make strong statistical inferences. The distance that is statistically significant at the 5% level is approximately equal to the inverse of the square root of the number of observations, thus more data is always better.

4. Data analysis

4.1 Deciding Data Categories

We analysed the data from four perspectives:

1. Human-oriented (H) versus techniques oriented (T) – to understand the focus. Which of the two is the most important perspective?
2. What should the university focus on? Based on a set of categories extracted from the data available – open coding [9].
3. Which requirements are implicit, which are explicit, and which are surprises
4. What competence is the industry missing when they employ our students

We decided what was human-related versus what was technology-related based on a simple rule: if it was related to being, we classified it as human and if it was related to doing, we classified it as technical.

When we had split the responses into human-related and technology-related responses, we read through all the requirements and assigned a category to each requirement using open coding. The categories were derived as follows:

1. All requirements were sorted alphabetically. This helps us to get an overview of requirements that are formulated in a similar way and thus helps us to create a consistent set of categories.
2. Remove all textual qualifiers, such as "be able to…" and "have experience with…"
3. The remaining phrases were organized into groups, giving the categories shown in Table 1, which were used to categorize the responses.

Table 1: Derived categories

Categories			
L	Learn new things	S	Software development
C	Communicate	CT	Computer technology
A	Adapt / cooperate	ST	Software tools
P	Problem-solving	D	Development process
I	Independence	CR	Client relationship
LE	Leadership	MA	Mathematics
HAW	Hard working	HE	Hands-on experience
NP	Nice person	BU	Business understanding

We had to adjust some of the results from applying this rule in some cases – e.g., "Company values, and the subjects they encompass" was first assigned to the category CR (client relationship) and then later to BU (business understanding). However, re-assignment of categories was rare – less than 5% of all requirements.

It is important to note that other ways of categorizing the data could have given other results. However, the categories chosen are considered useful since they allow us to group the data into meaningful categories and then use the result as a basis for a meaningful discussion. In order to clarify our choices, we show three responses for each category in Table 2 – a typical response and two responses that are in the "outskirts" of the category.

Table 2: Category definitions

Category	Typical requirement	The outer fringe – 1	The outer fringe – 2
L	eagerness to learn and always be learning	curiosity	always strive for a higher standard
C	communicate well	interpersonal skills	presentation
A	adaptable to new challenges and problems	respect his colleagues	not taking the first solution to a problem
P	Problem solving	able to keep an overview	able to think in abstract terms
I	independent	self-motivated	to take initiative
LE	ability to lead	arouse enthusiasm in others	takes responsibility
HAW	concentrate, and to focus on details	ability to handle a high workload	focus for extended periods of time
NP	empathy	enthusiasm	find enjoyment in what that you do/create
S	programming skills	general experience and understanding of programming	able to fix new problems and bugs
CT	concepts within hardware that are close to software	good at algorithms and maths	reason about algorithm complexity/system performance
ST	use established methods	knowledge about our preferred tools and techniques	experience with tooling

D	basic knowledge of systematic working methods	knowledge of industry standards	understand the project they are working on
CR	understand business and what it is the customer needs	grasp what the customers wants	ability to interact with customers and create solutions
MA	has an eye for applied mathematics	scientific approach	see the practical application in new theory
HE	practical experience	experience with larger projects and real costumers	work with a major project
BU	Business understanding	company values, and the subjects they encompass	having a general idea about the business side of things

As we will see later, the two most important categories are software development skills (S) and ability to adapt / cooperate (A). We have thus also used open coding to get a set of subcategories for each of these two as shown in Table 3.

Table 3: Subcategories for programming skills (S) and adaptability (A)

Category	Typical requirement - S	Category	Typical requirement - A
DB	database skills	TE	experience developing as a team
DES	recognize when to use different structures and design choices	FLE	work flexibly
GP	clean and readable code with good documentation	PR	work in a project environment
MT	experience, especially with maintaining code	HU	thinks that they know more than what they do
SEC	security	CO	cooperative
SPL	programming languages	PRO	adapting to a variety of problems and situations
ALG	understanding of algorithmic complexity and efficiency		
GCS	interested in computer science/software development		
OP	knowledge of basic Linux system administration		

We have not discussed all the categories that we have used. Instead, we have first checked that Pareto's rule (Pareto, 1906) applies – 70% of the effect stems from 30% of the causes. This will affect our discussions in two ways – both beneficial: we will focus on the few important factors and we will ignore a lot of not-so-important-factors except for the information that they are there and that they are not so important, at least not until the important factors are catered to.

To describe the companies that were interviewed, they were placed in one out of four categories – software (SW), hardware (HW), Research (RES), University (UNI) and Other (OTH). The 17 companies in the OTH category are companies that mainly use software to supply services. Most of their software is developed by others, even though they also develop some software on their own. Of the 17 companies, we find 11 that use software to sell services or develop mechanical products, four network operators and one consultancy company.

In the rest of the paper, we will compare the assigned importance of each category depending on their ranking. When we need to compare ranking from several domains, we will use the Spearman correlation coefficient.

4.2 Requirement Importance

To get a better idea of the importance of each requirement, we applied Kano's model for customer satisfaction [6]. The main idea of Kano's model is that product characteristics can be split into three categories – (1) obvious or hygienic, (2) required – what the customer explicitly wants and (3) surprises – something the customer did not expect but will appreciate – see Figure 1. The two first characteristics align well with the ISO 9001 definition of quality characteristics, which requires satisfaction of both implicit (hygienic) and explicit (required) requirements.

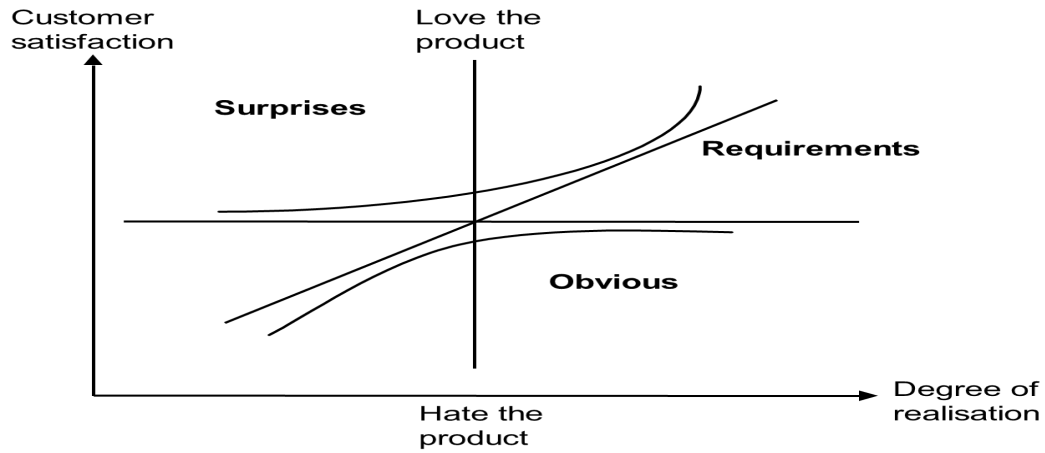

Figure 1: The Kano model of quality characteristics

Characteristics that fall into the "Obvious" category are often not specified, they are just assumed to be present. If they are missing, the customer will be strongly disappointed. Over time, requirements will move downwards in the model – from surprise to requirement and finally down to obvious, because, in the end, everybody is assumed to meet them.

Thus, requirements that end up as obvious ***mus***t be part of what the students know about while the requirements that end up in the surprise category will be the requirements that makes a student stand out from the crowd.

5. What did we find from the data analysis

5.1 Human Versus Technology

Of all the requirements put forward by the companies, 373 were related to human characteristics and 268 were related to technology. It is reasonable to see the human part of the categories as a description of the person you want to employ, while the technical characteristics is a description of what the person is able to do. The larger number of human-related requirements tells us that it is easier to change what people know than to change what they are. The distribution of human (H) and technical (T) requirements vary over the company categories as shown in the bar chart in Figure 2.

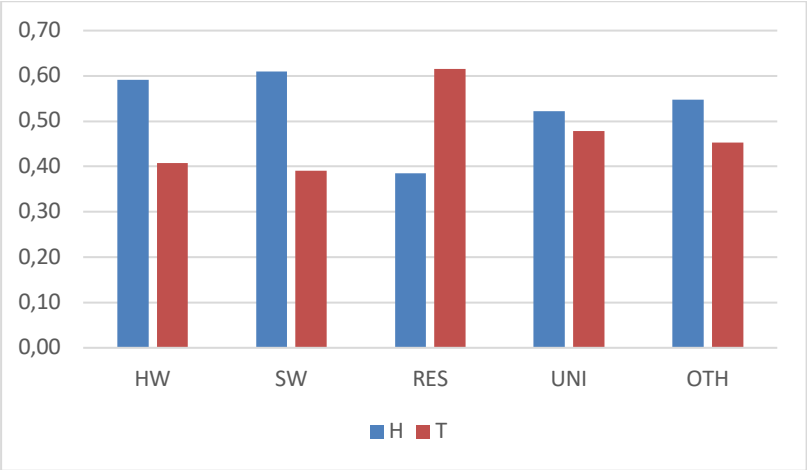

Figure 2: H and T requirement distribution for the five business domains

Only the research companies (RES) give a higher priority to technical than to human requirements. For HW, SW and RES, the differences between the distributions of human and technical requirements are statistically significant at the 5% level – around 20 percentage points. This result is in general agreement with what we found when looking at the related work.

5.2 Requirements distributed over derived categories

If we ignore the requirements that could not be assigned to any of the defined categories, we get the results shown in Table 4 below. As should be expected, software development is the most important category, followed by the ability to adapt and to learn new things. Note that categories such as software tools (ST) and development process (D) get little attention.

Table 4: Distribution of requirements over categories

Category (H)			**Category (T)**		
L	Learn new things	75	**S**	Software development	155
C	Communicate	63	**CT**	Computer technology	36
A	Adapt / cooperate	82	**ST**	Software tools	15
P	Problem-solving	51	**D**	Development process	18
I	Independence	23	**CR**	Client relationship	22
LE	Leadership	31	**MA**	Mathematics	7
HAW	Hardworking	28	**BU**	Business knowledge	4
NP	Nice Person	11			
none					20

If we, instead of counting occurrences, count number of companies that require competence within a certain category, we get the diagram shown in Figure 4 below. The four most important categories are S, A, L and C.

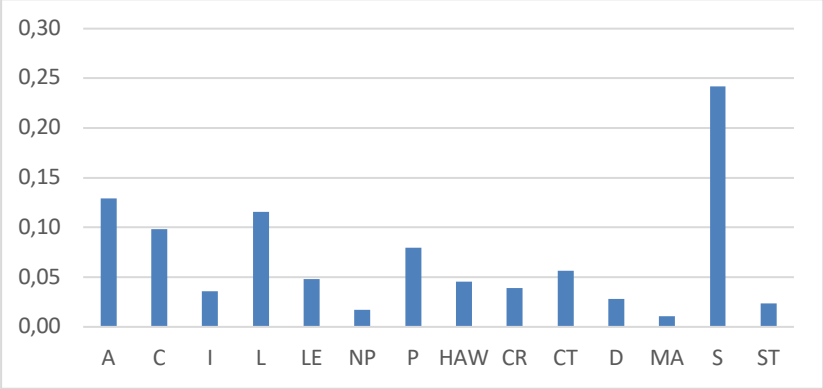

Figure 3: Percentages of requirement categories

We see that 30% of the categories (S, A, L, C and P) accounts for 76% of all requirements. Thus, Pareto's law [7] holds.

If we consider each company type separately, HW, SW etc. – we get the diagram shown in Figure 4. In order to get an easily readable graph, we have left out two datasets with few data – data for research institutes (RES) and universities (UNI).

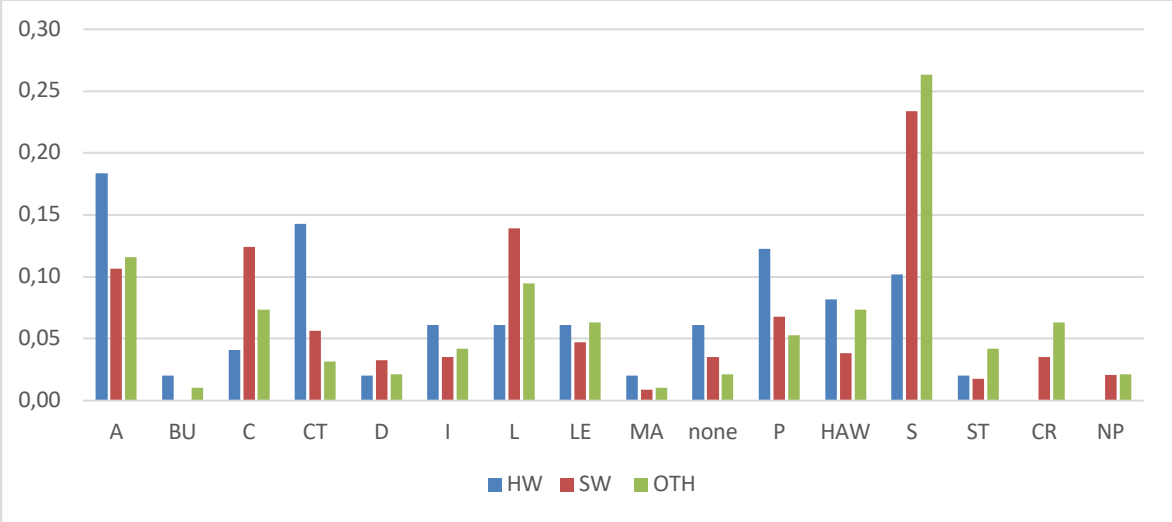

Figure 4: Required competences in percentages for companies in selected domains

The relative importance varies between the domains. For software, the three most important categories are Software development (S), Learn new things (L) and Communication (C). For hardware, the most important characteristics are to be Adaptable (A), Computer Technology (CT) and Problem solving (P). For the OTH domain, the three most important characteristics are software development (S), being adaptable (A) and the ability to learn new things (L).

The agreement between the domains when it comes to the importance ranking of the categories vary quite a lot, as shown by the Spearman correlation, e.g.:

- HW – SW: 0.73, p = 0.002
- SW – OTH: 0.86, p = 0.000
- HW – OTH: 0.54, p = 0.030

5.3 What Will Make a Candidate Stand Out From The Crowd

We split the data according to whether they were human related (H) or technology related (T) and then applied Kano's model (Kano, 1984) to decide whether the requirements were hygienic (1), required (2) or surprising (3). These categories were assigned to each requirement based on the interviewer's impression. Not all the companies assigned this category to their data during the interviews. Thus, there are less data here than in the previous table. The analysis gave us the results shown in Table 5.

Table 5: Human related and technology related characteristics split according to Kano's model – absolute values

Category	**H1**	**H2**	**H3**	**Category**	**T1**	**T2**	**T3**
Learn new things	30	38	2	Software development	62	43	5
Communicate	11	42	3	Computer technology	14	7	6
Adapt / cooperate	19	51	1	Software tools	8	5	0
Problem-solving	3	17	13	Development process	7	3	2

Independence	1	3	4	Client relationship	3	1	13
Leadership	9	8	8	Mathematics	1	0	3
Hardworking	8	9	0	Business knowledge	0	0	3
Nice Person	8	0	0				
none	15						

When it comes to human requirements (H), the three most important hygienic categories are Learn new things (L), Communication (C) and Adapt / cooperate (A). These three characteristics are also among the four that are most important in the requirements – see Figure 3. Being good at problem solving makes a candidate stand out from the rest.

For technology requirements (T), software development is the most important thing both as expected and required characteristic. Being good at handling customers is a welcome surprise that again makes a candidate stand out from the crowd.

5.3 What is important for the industry

Table 6 shows the part of the scores for each category and for the four most important domains. We will focus on the top five categories – 30%. If we sum up the five categories with the highest scores for the three categories, we get 0.62 (HW), 0.67 (SW) and 0.61 (OTH) respectively.

Adaptability (A) and software development (S) are among the five with the highest score in all cases. Computer technology (CT) is mentioned only once – for the hardware domain – while problem solving (P), learning new things (L) and communication (C) are among the top five in two out of three domains. Being hardworking (HAW) is not among the top five for SW.

Table 6: Scores for each category for three important domains

HW		**SW**		**OTH**	
Category		**Category**		**Category**	
A	0,18	S	0,23	S	0,26
CT	0,14	L	0,14	A	0,12
P	0,12	C	0,12	L	0,09
S	0,10	A	0,11	C	0,07
HAW	0,08	P	0,07	HAW	0,07
I	0,06	CT	0,06	LE	0,06
L	0,06	LE	0,05	CR	0,06
LE	0,06	HAW	0,04	P	0,05
none	0,06	I	0,04	I	0,04
C	0,04	none	0,04	ST	0,04
BU	0,02	CR	0,04	CT	0,03
D	0,02	D	0,03	D	0,02
MA	0,02	NP	0,02	none	0,02
ST	0,02	ST	0,02	NP	0,02
CR	0,00	MA	0,01	BU	0,01
NP	0,00	BU	0,00	MA	0,01

During the interviews, the respondents were asked about both the persons' requirements – the "what" – and why each requirement was important – the "why". Out of the 640 requirements, 370 requirements (58%) were accompanies by a "why". The why-percentage was approximately the same for all application domains – 59% to 63%.

For adaptability (A), the two whys "adapt to the team's way of working" and "adapt to new methods" make up 80% of all whys. For software development (S), the three whys are: because we need (1) "development languages and design", (2) "general software development competence" and (3) "new ways to develop software" These three make up 82% of all whys for software development.

6 What the industry is missing

6.1 Missing Competencies

The question was "Q3: Based on your experience, which skill(s) – if any – are typically lacking from students graduating from University or College". Using the same categories as the ones used for requirements, we get the diagram shown in Figure 5.

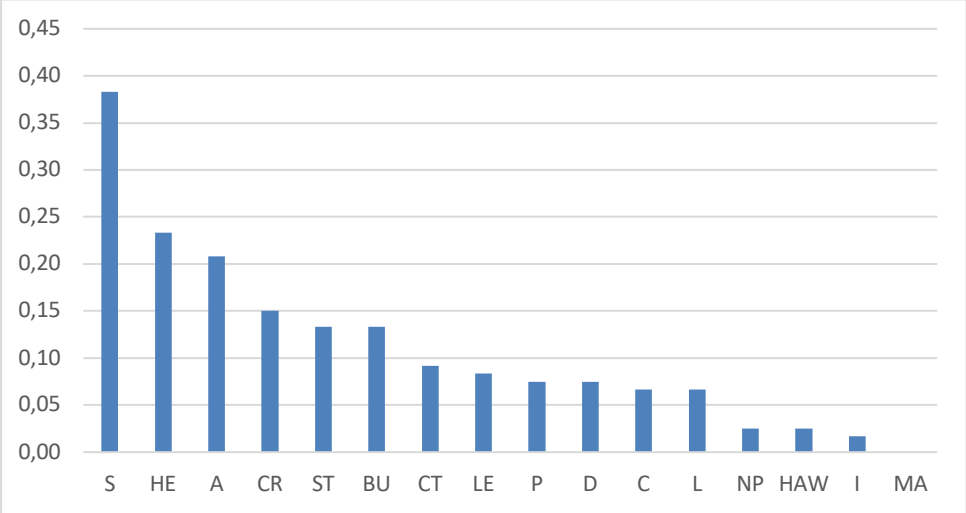

Figure 5: Missing competences in percentages of companies

As can be seen in Figure 10, the important but, alas, missing competences varies over the application domains.

- SW: programming skills, adapt / cooperate and hands-on experience
- HW: programming skills, business understanding and adapt / cooperate.
- OTH: programming skills, hands-on experience and adapt / cooperate,

Independence (I), mathematics (MA) or being a nice person (NP) are neither required nor missed for any of the company categories.

If we use percentages for each company category and remove the two categories with only a few requirements (10 and 12 respectively) we get the diagram shown in Figure 6.

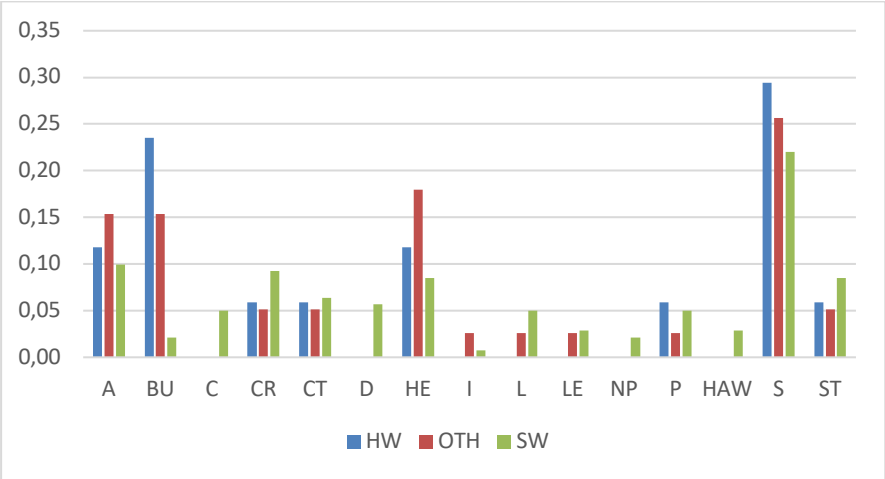

Figure 6: Company wise distribution of missing competences

The requirements categories programming skills, hands-on experience and the ability to adapt / cooperate are the most frequently mentioned missing competences for all of the domains. The SW domain is different from the two others since it has Client Relationship (CR) as one of the top four, HW and OTH have Business knowledge (BU) among the top four instead. The top four missing competences make up 77% and 74% for HW and OTH but only 50% for SW.

When we split the missing categories according to domain, we see that there is a lot of disagreement when it comes to ranking the categories as shown by Spearman correlation.

- HW – SW: 0.59, p = 0.016
- SW – OTH: 0.56, p = 0.002
- HW – OTH: 0.90, p = 0.000

The rankings for HW and OTH are in almost complete agreement.

6.2 Derived Categories for S and A

Using the same open coding techniques as we used previously, we assigned subcategories to the two most frequently mentioned competencies – software development (S) and adaptability (A). The chosen categories plus an example for each is shown in Table 3.

For software development (S), the distribution of answers according to the categories used in Table 3 are shown in Figure 7. We see that general programming knowledge and maintenance are the top two items.

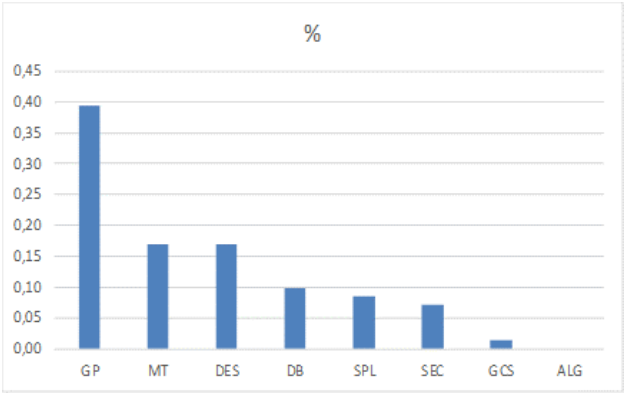

Figure 7: Software development categories (S)

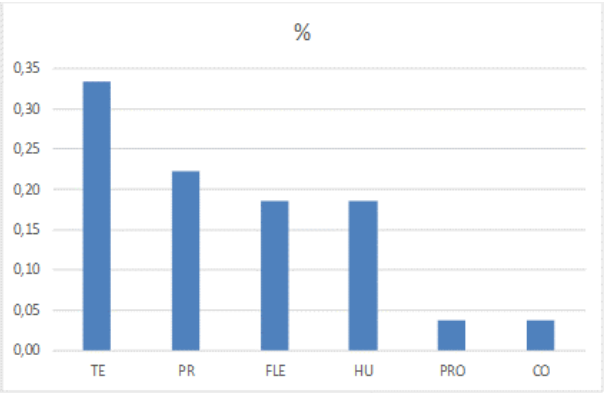

Figure 8: Adaptability categories (A)

The other high-score category – adaptability (A) – is distributed as shown in Figure 8. The categories used are explained in Table 3. The most important part of adaptability is to have experience with working in teams, followed by working in a project environment.

For software development (S), the distribution of answers according to the categories used in Table 3 are shown in the diagram below. We see that general programming knowledge and maintenance are the top two items. The most important part of adaptability is to have experience with working in teams, followed by working in a project environment.

7 Threats to validity

The large number of companies interviewed (120) and the spread in areas of work makes the company sample large enough to allow reasonable generalizations and statistical inferences at the 5% level of significance. The most important threats to validity are the definitions of categories and the assignment of categories to each company requirement.

We need to keep in mind that the texts used to assign all the categories used are rather wide. Some of the requirements are so vague that they probably could fit into two or three categories – see Table 2 for examples. This problem is related to the use of open coding. When the coding category was decided, we coded the data material. This introduces several new challenges, e.g., requirements that cover several challenges such as "understand business goals and what the customer needs", which could be either BU or CR.

An important source of threats to repeatability is the fact that besides the three questions quoted in chapter 3, there was no interview guide. Thus, different persons may have had different interpretations of the same question. However, the uniformity of the responses, both for requirements and for missing requirements indicates to us that most of the interviewers have gotten it right.

Last, but unfortunately not least, there is the problem identified by J. Kano (1984). Some requirements that are not mentioned, may have been left out because the respondents thought they were obvious. E.g., there is nowhere mentioned that the employee must be able to read and write – these are obvious requirements. However, we cannot know how many other requirements that are left out for the same reason. Thus, we should take care when we make statements based on the question "Why isn't X here – that is always needed".

8 Some tentative conclusions

For all companies, there are more requirements related to what people are than to what people know. For HW, SW and RES, the differences are statically different at the 5% level. If we consider all responses together, the five categories software development (S), adaptability (A), learning new things (L) and communication (C) makes up 76% of all requirements. All of the three domains HW, SW and OTH have adaptability (A) and software development (S) among the top five requirements. Except for HW, the two other domains also have learning new things (L) and communication (C) among the top five. Only OTH has hard working among the top five requirements. Applying Kano's model of quality characteristics, we found only two requirements in the "surprise" category: problem-solving and client relationship.

Concerning what the industry feel is lacking, the top four categories are software development (missed by 38%), hands-on experience (missed by 23%), adaptability (missed by 21%) and client relationship (missed by 15%). When "software development" (S) is missed, this does not mean that candidates know nothing about it (which should be impossible when graduating with an IT major), but that they know too little. The two major issues are too weak general programming competence – i.e., coding (40%) and limited knowledge about maintenance – both writing maintainable code and doing maintenance on existing code (16%). Similarly for lacking adaptability (A) of the candidates, two notable issues on the more detailed level were weak team-work skills, skills, i.e. problems adapting to the other persons in the team (34%) and weak project skills, i.e. trouble adapting to the way the company run projects (22%).

On aggregate, human treats appear more important than technical competence. Yet, technical competence cannot be neglected, as responses about shortcomings indicate that candidates should be stronger in practical software development.

9 How can our degree programs be improved

Going beyond the direct findings from the study, it is interesting to ask how degree programs in IT can be improved, especially to better address some of the aspects that industry found less satisfactory. Improvement could be attempted in many different ways, ranging from small scale changes (e.g., new content, learning methods, and assessment approaches in existing courses) to large scale changes (e.g., massively altering the structure and learning approach of entire degree programs). In the following, we

will discuss more concretely first the identified shortcomings in technical competence, then the soft skills.

Coding, maintenance, hands-on experience. Most IT degree programs in Norway have not just one but several courses in programming, from the freshman year (Lorås et al., 2018) and onwards. Yet, as our interview results indicate, coding skills of graduates do not always live up to industry expectations. First, it might be that the shortcomings apply to some candidates, but far from all. There is high variation in students' mastery of coding. Projects – which may offer the most hands-on coding experience – are often done in teams, sometimes with the effect that most of the coding is done by the students who are already good at it, while others focus on alternative tasks (e.g., documentation, report writing, presentations). Second, there are differences between how coding is taught, practiced, and assessed at university, and how real coding work is done in industry. This also relates to the lack of maintenance competence among university graduates. Software development at university is typically taught in a *greenfield* development context (Fox and Patterson, 2012, Szabo, 2014) – i.e., making code from scratch – and seldom involves maintenance of legacy code, which is more common in industry and a typical task for fresh employees. Three possible action points for IT degree programs at the NTNU are therefore: (1) Give students who have fallen behind in coding after the initial courses an opportunity to catch up with their peers, rather than lagging further behind. In particular, for student team projects, require a more equal distribution of various types of tasks among students, to prevent that only the best coders do the coding work. (2) Find ways to focus more on maintenance in coding and software development courses. According to the principles of constructive alignment (Biggs, 2014), it is important that such maintenance tasks are also included in the assessment, counting towards the grade. There are papers suggesting interesting approaches to address maintenance more, such as having students do improvement of open source software (Ellis et al., 2007), or maintenance of legacy software of non-profit organizations (Fox and Patterson, 2012).

Soft skills: client relationship, adaptability. When it comes to shortage of soft skills, for instance communication skills needed for relating to clients, the problem may be somewhat similar to that of coding: Some students have good communication skills, others less so. Again, team projects are one of the main arenas for hands-on training of communication tasks such as client interviews and presentations, and students will often tend to distribute tasks such that those who already have good communication skills end up doing the most communicative work in the project. As for the shortcomings related to adaptability, some revealing quotes from the interviews: "[recent graduates] …think that they know more than they do" and are "…unable to adjust to a group". The latter issue might be surprising, as there is quite a lot of group work in our IT study programs. However, group work in university is different from industry in several important ways: Student groups tend to be rather homogeneous – students are all in the same study-year, taking the same course, thus often belonging to the same study program, too. One exception from the latter is NTNU's Experts in Team project, which is interdisciplinary, so that each team will have students from several different study programs. However, the students are all on the same level, with symmetric roles (all being in the project as "experts" in their own discipline), and all having a say in how to run the project and what to deliver. Hence, the amount of necessary adaptation is much smaller than what it might be in industry, where the fresh employee enters a project team as the "rookie" and must comply with already established working practices decided by others. Possible action points concerning these soft skills: (1) ensure that all students get sufficient exposure to learning activities addressing communication skills, such as client interviews and presentations – rather than just some few students that choose to take such roles in project teams. (2) Expose the students to some software projects where they are put in a team with more experienced people, rather than always with peers on the same level as themselves, so that they get practice adapting to a more diverse set of co-workers. One possible way of doing this might be through bigger focus on placements, internships, etc., another to have more student projects

collaborating across different years of study, different disciplines, different campuses, or even between different countries, sometimes with an asymmetric relation between project participants (e.g., solution architects vs subcontractors) to expose the students more to more diverse collaboration settings.

"The Times They are A-Changin." A decade or two ago, industry tended more to emphasize specific tools and methods when asked about lacking competence of candidates. Our interview material indicates that employers have now understood that methods and tools are often short-lived. Hence, ability to learn *new* methods and tools may be more important than the knowledge of particular tools. While university *is* about learning new things, the focus will sometimes be more on new scientific theory than on new methods and tools for practitioners. To some extent there is a trade-off between academic ambitions (e.g., of professors, who would like more specialization in the course ladder, to get high synergy between their research and teaching and prepare students for doing brilliant master thesis research and perhaps continue with PhD studies) and satisfying industry – who more often want broad generalists with hands-on experience (e.g. knowing a bit about databases, operating systems, software engineering, security, user interfaces, artificial intelligence, parallel computing …) rather than narrow specialists who know a lot about 1-2 of these topics but little about the rest.

The challenges faced by NTNU's Computer Science department could be seen as related to the model suggested by Archer (1982). She claims that all academic education goes through three phases:

1. Take-off: The contents of the education is close to what the practitioners do.
2. Growth: the contents is gradually becoming more independent of what the practitioners do.
3. Inflation: the education carries on with its independent life – decoupled from the practitioners' ways of doing things.

Phase 1 comprised the early years (for Computing meaning 70's, 80's, partly 90's). A gradual drift into phase 2 could be associated with the shift from being a technical university (NTH) to a general university (NTNU, est. 1996). A strong emphasis on academic standing rather than industrial impact might lead to phase 3. In particular, faculty from previous colleges that merged with NTNU in 2016 are worried that their study programs, with a more hands-on focus, shall be over-academized in the new university setting.

An additional reflection upon Archer's model, however, is that "growth" is not necessarily due to a drift away from industry needs, but partly a result of growth of the practice field itself. This is especially true for computing, with its rapid development. Over the last 20-30 years, the diversity of devices and usages of computing has exploded, as has the development frameworks and approaches. Every year there are new topics – also industry-relevant ones – which would be interesting to include in a study program, but not a similar list of existing courses that can easily be removed.

Another point of discussion is how far universities should go to satisfy the wishes of employers. Some graduates shall not apply for jobs, rather become entrepreneurs who make their own companies. Some competencies will necessarily be more effectively learnt at work, for instance domain specific knowledge and skills. As discussed in (Tholen, 2018) it will be quite costly for a university to set up courses and projects that closely mimic "real-world" situations, while in the work place this comes for free, since they ***are*** the "real-world".

Acknowledgement
The data was collected by students as part of their course-work in "TDT4140 Software Engineering", taught by professor Pekka Abrahamsson. Without his effort, and that of his teaching assistants and the students themselves, the research behind this article would not have been possible.

References

AGUILA, G. M., DE CASTRO, E. L., DOTONG, C. I. & LAGUADOR, J. M. 2016. Employability of computer engineering graduates from 2013 to 2015 in one private higher education institution in the Philippines. *Asia Pacific Journal of Education, Arts and Sciences,* 3**,** 48-54.

AHMED, F., CAPRETZ, L. F. & CAMPBELL, P. 2012. Evaluating the demand for soft skills in software development. *It Professional,* 14**,** 44-49.

ARCHER, M. S. 1982. *The sociology of educational expansion: Take-off, growth and inflation in educational systems*, Sage Publications.

BENAQUISTO, L. & GIVEN, L. 2008. The SAGE encyclopedia of qualitative research methods. *New York: Sage*.

BIGGS, J. 2014. Constructive alignment in university teaching. *HERDSA Review of higher education,* 1**,** 5-22.

DEAK, A. & SINDRE, G. 2013. Analyzing the importance of teaching about testing from alumni survey data. *Norsk informatikkonferanse (NIK),* 2014.

ELLIS, H. J., MORELLI, R. A., DE LANEROLLE, T. R. & HISLOP, G. W. Holistic software engineering education based on a humanitarian open source project. 20th Conference on Software Engineering Education & Training (CSEET'07), 2007. IEEE, 327-335.

FALLOWS, S. & STEVEN, C. 2000. Building employability skills into the higher education curriculum: a university-wide initiative. *Education+ training,* 42**,** 75-83.

FINCH, D. J., HAMILTON, L. K., BALDWIN, R. & ZEHNER, M. 2013. An exploratory study of factors affecting undergraduate employability. *Education+ Training,* 55**,** 681-704.

FOX, A. & PATTERSON, D. 2012. Crossing the software education chasm. *Communications of the ACM,* 55**,** 44-49.

HAMILTON, M., CARBONE, A., GONSALVEZ, C. & JOLLANDS, M. Breakfast with ICT Employers: What do they want to see in our graduates. Proceedings of the 17th Australasian Computing Education Conference (ACE 2015), 2015. 30.

HEALEY, M. & JENKINS, A. 2009. *Developing undergraduate research and inquiry*, Higher Education Academy York.

KANO, N. 1984. Attractive quality and must-be quality. *Hinshitsu (Quality, The Journal of Japanese Society for Quality Control),* 14**,** 39-48.

KITCHENHAM, B., BUDGEN, D., BRERETON, P. & WOODALL, P. 2005. An investigation of software engineering curricula. *Journal of Systems and Software,* 74**,** 325-335.

KOLÅS, L. & MUNKVOLD, R. I. 2017. Learning through construction: a roller coaster ride of academic emotions? *Proceedings of the 6th Computer Science Education Research Conference.* Helsinki, Finland: ACM.

LAUVÅS, P. & RAAEN, K. 2017. Passion, Cooperation and JavaScript: This is what the industry is looking for in a recently graduated computer programmer. *UDIT 2017.* Oslo.

LETHBRIDGE, T. C. 2000. What knowledge is important to a software professional? *Computer,* 33**,** 44-50.

LORÅS, M., SINDRE, G. & AALBERG, T. 2018. First year computer science education in Norway.

LUNDBERG, G. M., GAUSTAD, A. & KROGSTIE, B. R. The employer perspective on employability. 2018 IEEE Global Engineering Education Conference (EDUCON), 2018. IEEE, 909-917.

MARKES, I. 2006. A review of literature on employability skill needs in engineering. *European Journal of Engineering Education,* 31**,** 637-650.

MARKS, A. & SCHOLARIOS, D. 2008. Choreographing a system: skill and employability in software work. *Economic and Industrial Democracy,* 29**,** 96-124.

PARETO, V. 1906. *Manuale di economia politica*, Societa Editrice.

SURAKKA, S. 2005. Analysis of technical skills in job advertisements targeted at software developers. *Informatics in Education-An International Journal,* 4**,** 101-122.

SZABO, C. Student projects are not throwaways: Teaching practical software maintenance in a software engineering course. Proceedings of the 45th ACM technical symposium on Computer science education, 2014. ACM, 55-60.

THOLEN, G. 2018. The limits of higher education institutions as sites of work skill development, the cases of software engineers, laboratory scientists, financial analysts and press officers. *Studies in Higher Education***,** 1-12.