A study on constructing capability indicators for C++ learners at universities in Taiwan

David W.S. Tai†, Jia-Ling Chen‡, Ren-Cheng Zhang‡, Ray Wang† & Zena Y.H. Tai*

Hungkuang University, Taichung City, Taiwan†
National Changhua University of Education, Changhua, Taiwan‡
National Yunlin University of Science and Technology, Douliu City, Taiwan*

ABSTRACT: The main purpose of this study was to establish appropriate capability indicators for programming design learners at universities in Taiwan. In the article, the authors describe and discuss the final results by using a modified Delphi method and the analytic hierarchy process (AHP) approach, involving 30 experts from academia and industry. Furthermore, three probes of the modified Delphi technique were conducted and 39 capability indicators were proposed after the completion of this process. The results indicate that *Repeating Flow Control*, *Logic Decision* and *System Function* are the most important indicators in programming design capability. However, the experts from industry thought differently from the academic experts.

INTRODUCTION

The main educational purpose of information engineering is to develop the capacity of students to achieve the human resource (HR) requirements of the market, especially, in the information management domain. Therefore, in curriculum design, both the theories and the practices need to be considered as they should enrich the complementary function between academia and industry. On the other hand, preventing HR supply and demand difficulties from happening in industry is an important issue. For this reason, development towards an industry orientation is required; however, many universities cannot afford the appropriate laboratory equipment to develop the required practical skills [1].

In order to harmonise university education with industry needs, a programming language is needed to construct a set of capability indicators, but the curriculum design process in engineering education lacks feedback from industry, experts and graduates. Therefore, these institutions should verify the demands of the industry, and plan a set of appropriate curricula to cater to those needs. To students who major in computer related subjects, the programming design ability is one of the core competencies, and it is also important to other engineering students [2].

When graduates enter the job market, they face the diverse environment of the software industry. Therefore, educators and students should be well prepared to confront these challenges. In the meantime, it is necessary to create an efficient educational environment for different types of students [3]. Consequently, academia and industry should cooperate to change students' attitudes and learning motivation, and to face challenges such as skills, knowledge and professional needs [4].

In this field, the programming language ability is one of the most important prerequisites to enter the market. C++ is particularly well applied in the software industry, and the main purpose of this study was to investigate and verify the importance and differences of the capability indicators between academia and industry.

LITERATURE REVIEW

Programming Design

In ordinary universities or universities of science and technology, programming design is one of the required courses in information related departments. Additionally, it is a fundamental course for most students in their future careers. More specifically, this skill is a core ability needed by many students in this field, such as engineering and computer science [5], and learning programming design should be practical and specific [6].

Many programming design instructors have stated that teaching programming design is not just about teaching a specific programming language, but it involves teaching students how to write a program [7]. Moreover, learning programming design includes several activities, such as program characteristics, design and comprehension.

Typical instruction begins by revealing related information and knowledge of a specific programming language, but this is not sufficient. The common principles of designing a programming language are important, as they guarantee the quality of the software. However, beginners in programming design often omit them and, therefore, learning feedback is not provided to those students [8].

Capability Indicator Construction

The development of professional capability in professional subjects is often a major challenge, and evaluators should make sure that the evaluation methods can cover all the knowledge and skills that should be rated during the process [9]. In order to ensure professional capability, a questionnaire or survey can be used to collect the opinions of the experts and employees, and to rank the importance of the various capabilities. Evaluation methods include functional analyses, OCAPs, DACUM, V-TECS, Delphi and Fuzzy Delphi [10].

Among the above mentioned methods, the Delphi method is usually used to identify research problems, proceeding from related plans, as well as evaluating and developing existing proposals [11]. Moreover, the Delphi method has been shown to be an ideal way to cohere to a common point of view from different experts, and it is anonymous and private compared with other ways, such as expert panels [12].

In addition, the results of the Delphi method are often used to analyse the work demands in a job market, which can include educational training, evaluation, human resource appraisal, job reforms and human resource planning. Using an effective analysis method can distinguish job contents, required skills, knowledge, attitudes and job specification; and based on the results, industries and educational training institutions can design appropriate teaching materials, to ensure the integrity of indicators.

On the other hand, the analytic hierarchy process (AHP) can be used to construct the comparative importance between indicators. This method was developed by T.L. Saaty in 1971, and it can help decision-makers to clarify complex circumstances by using hierarchy [13]. Moreover, it can provide other methods and rank them when the decision-maker needs to choose the most appropriate solution. Through this characteristic, it is possible to calculate the comparative importance of the various programming design indicators, and to know which capability indicator is the most important ability.

METHOD

The study used the expert panel, a modified Delphi method and AHP approach to conduct the basis capabilities of programming design. The main goal of using the modified Delphi method was to collect the opinions of 30 industrial and the academic experts in the information related areas. Furthermore, three probes of the modified Delphi technique were conducted and 39 capability indicators were proposed after this process. There should be at least five to ten indicators from different professional groupings [14]. Overall, there were 29 participants in the whole process, of which 23 were from academia and six from industry, as shown in Table 1. The effective response rate was 96.6. Therefore, it matched the assumption.

Table 1. Delphi groupings and number of participants.

Type	No.	Percentage
Academia	23	79%
Industry	6	21%
Sum	29	100%

This study used content analysis to verify the opinions and suggestions from the experts according to Probe 1. These were, then, categorised into the same or similar opinions, and compared with different opinions as a reference material to Probe 2.

Table 2. Timetable of the Delphi probes.

Round	Probe	Date	No. of participants	Valid	Response rate
1	Probe 1	6 March 2012	30	30	100%
2	Probe 2	16 April 2012	30	29	97%
3	Probe 3	14 May 2012	29	29	100%

RESEARCH FINDINGS

Delphi

The data processing criteria were established according the principles below:

1. The means of importance is over/equal to 3.5. It indicates that the item is important.
2. The quartile deviation is less than 1 or the SD is less/equal to 1. It indicates that the experts have the same viewpoints.
3. The principles for removing the items: the means of the importance is less than 3.5, and the quartile deviation is over 1.0.
4. The Kolmogorov-Smirnov one sample test for goodness of fit was used to ensure the trends that the experts agree with in every indicator.
5. The Kruskal-Wallis one-way analysis of variance by rank was used to test the consistency between the experts from academia and industry. This is non-parametric statistics and to test the equalisation of the medians in independent population of non-normally distributed groups. Additionally, the samples in each group should be over 5 [15][16].

According to Probe 1 of the Delphi method, overall the indicator means are over 3.5, and it means the experts agreed the indicators are fairly important (see Table 3).

Table 3. Results of Delphi Probe 1 (N = 30).

Dimension	Indicator	Mean	SD	Quartile Deviation
1. Basics of C++	A. Introduction to programming	4.367	1.033	0.5
	B. Introduction to C language	4.333	0.661	0.5
	C. Data types and identifier	4.633	0.669	0.5
	D. Constant and variable	4.633	0.556	0.5
	E. Expression	4.633	0.669	0.5
	F. Input and output	4.600	0.621	0.5
2. Flow control	A. Logic decision	4.833	0.461	0
	B. Select flow control	4.733	0.521	0.125
	C. Repeat flow control	4.633	0.615	0.5
	D. Interrupt and break flow	4.367	0.765	0.5
3. File and application	A. Random access file	4.033	0.999	1
	B. Sequential access file	4.033	0.890	0.5
	C. Index access file	3.933	1.048	1
4. Subprogram and application	A. System function (subprogram)	4.600	0.675	0.5
	B. User-defined function (subprogram)	4.733	0.450	0.5
	C. Recursive function (subprogram))	4.133	0.776	0.5
5. Array and application	A. One dimension array	4.867	0.571	0
	B. Two dimensions array	4.833	0.592	0
	C. Multi dimensions array	4.033	0.928	0.5
	D. String	4.767	0.430	0.125
	E. Stack	3.967	1.159	1
	F. Queue	3.933	1.143	1
	G. Searching	4.333	0.959	0.5
	H. Sorting	4.467	0.819	0.5
6. Linked list and application	A. Pointer	4.600	0.675	0.5
	B. linked list	4.200	1.031	0.5
	C. Stack	4.000	1.083	1
	D. Queue	3.967	1.066	1
	E. Searching	4.200	1.031	0.5
	F. Sorting	4.133	1.074	1
7. Class and application	A. Structure	4.600	0.563	0.5
	B. Union	3.833	1.117	0.625
	C. Enumeration	4.000	0.743	0
	D. Class	4.733	0.521	0.125
8. Object and application	A. Object	4.833	0.379	0
	B. Inheritance	4.700	0.535	0.5
	C. Encapsulation	4.633	0.615	0.5
	D. Dynamic linking	4.033	0.890	0.625

Results of Delphi Probe 2 and Probe 3 are shown in Table 4.

Table 4. Results of Delphi Probes 2 and 3 (N = 29).

Dimension	Indicator	Probe 2		Probe 3		K-S	Probe 3				K-W
							Academia		Industry		
		Mean	SD	Mean	SD		Mean	SD	Mean	SD	
1. Basics of C++	1.1 Introduction to programming	4.45	0.910	4.52	0.688	0.131	4.57	0.662	4.33	0.816	1.009
	1.2 Introduction to C language	4.34	0.670	4.41	0.628	0.131	4.52	0.511	4.00	0.894	3.529
	1.3 Data types and identifier	4.83	0.384	4.83	0.384	0.000	4.83	0.388	4.83	0.408	0.003
	1.4 Constant and variable	4.72	0.591	4.79	0.491	0.131	4.83	0.388	4.67	0.816	0.000
	1.5 Operator and expression (modif.)	4.79	0.491	4.79	0.491	0.000	4.83	0.388	4.67	0.816	0.031
	1.6 Input and output	4.83	0.468	4.86	0.441	0.131	4.96	0.209	4.50	0.837	6.837**
2. Flow control	2.1 Logic decision	4.90	0.310	4.90	0.310	0.000	4.96	0.209	4.67	0.516	3.443
	2.2 Select flow control	4.90	0.310	4.90	0.310	0.000	4.96	0.209	4.67	0.516	3.443
	2.3 Repeat flow control	4.86	0.351	4.93	0.258	0.263	4.96	0.209	4.83	0.408	0.641
	2.4 Interrupt and break flow	4.41	0.568	4.34	0.553	0.263	4.30	0.559	4.50	0.548	0.627
3. File and application	3.1 Random access file	4.34	0.670	4.28	0.649	0.263	4.22	0.600	4.50	0.837	1.967
	3.2 Sequential access file	4.17	0.711	4.07	0.530	0.525	4.00	0.426	4.33	0.816	4.332*
	3.3 Index access file	3.97	0.944	3.93	0.884	0.263	3.91	0.900	4.00	0.894	0.219
4. Subprogram and application	4.1 System function (subprogram)	4.83	0.384	4.86	0.351	0.131	4.83	0.388	5.00	0.000	0.585
	4.2 User-defined function (subprogram)	4.86	0.351	4.93	0.258	0.263	4.91	0.288	5.00	0.000	0.065
	4.3 Recursive function (subprogram))	4.34	0.769	4.41	0.780	0.263	4.26	0.810	5.00	0.000	11.123**
5. Array and application	5.1 One dimension array	4.93	0.371	4.97	0.186	0.131	5.00	0.000	4.83	0.408	7.801**
	5.2 Two dimensions array	4.83	0.384	4.83	0.384	0.000	4.91	0.288	4.50	0.548	6.217*
	5.3 Multi dimensions array	3.90	0.772	3.93	0.799	0.131	3.96	0.825	3.83	0.753	0.553
	5.4 String	4.86	0.351	4.93	0.258	0.263	4.91	0.288	5.00	0.000	0.065
	5.5 Searching	4.28	1.099	4.34	0.974	0.131	4.39	0.891	4.17	1.329	0.000
	5.6 Sorting	4.38	0.903	4.41	0.867	0.131	4.39	0.891	4.50	0.837	0.231
6. Linked list and application	6.1 Pointer	4.66	0.670	4.59	0.825	0.131	4.65	0.714	4.33	1.211	0.018
	6.2 linked list	4.24	0.872	4.28	0.960	0.263	4.30	0.926	4.17	1.169	0.081
	6.3 Searching	4.31	0.930	4.31	1.004	0.131	4.35	0.982	4.17	1.169	0.037
	6.4 Sorting	4.31	0.930	4.17	1.037	0.263	4.17	1.029	4.17	1.169	0.022
	6.5 Dynamic and static memory allocation (added)	4.07	0.961	4.03	0.906	0.263	4.00	0.953	4.17	0.753	0.305
7. Class and application	7.1 Structure	4.52	0.574	4.69	0.471	0.525	4.65	0.487	4.83	0.408	1.116
	7.2 Union	3.86	0.875	3.86	0.789	0.263	3.70	0.765	4.50	0.548	13.589***
	7.3 Enumeration	3.93	0.651	3.93	0.458	0.263	3.83	0.388	4.33	0.516	8.274**
8. Object and application	8.1 Introduction to class (added)						4.52	0.665	4.33	0.516	0.838
	8.2 Class and object (modified)	4.83	0.468	4.86	0.441	0.131	4.83	0.491	5.00	0.000	2.034
	8.3 Inheritance	4.79	0.491	4.83	0.468	0.131	4.78	0.518	5.00	0.000	2.718
	8.4 Encapsulation	4.62	0.728	4.76	0.511	0.263	4.74	0.541	4.83	0.408	0.335
	8.5 Dynamic linking	4.24	0.912	4.17	0.928	0.131	4.09	0.949	4.50	0.837	3.370
	8.6 Polymorphism (added)	4.41	0.983	4.41	0.867	0.263	4.26	0.915	5.00	0.000	8.005**
	8.7 Virtual function (added)	4.17	0.848	4.07	0.884	0.263	3.91	0.848	4.67	0.816	12.294***
	8.8 Exception (added)	4.34	0.769	4.38	0.728	0.131	4.22	0.736	5.00	0.000	12.636***
	8.9 Pattern (added)	3.97	0.981	3.97	0.906	0.263	3.87	0.920	4.33	0.816	3.602

*p < 0.05 **p < 0.01 ***p < 0.001

As shown in Table 4, one indicator was modified (1.5 Operator and expression) and five indicators added (6.5 Dynamic and static memory allocation; 8.6 Polymorphism; 8.7 Virtual function; 8.8 Exception; 8.9 Pattern). In the second probe, all the indicator means are over 3.5 as well, and it shows the experts agreed with the opinions that came from the first probe, and the viewpoints from different experts tended to be stable in this stage.

However, two indicators (8.1 Introduction to class; 8.2 Class and object) were proposed in response to the second probe. In the last probe, all the indicator means are also over 3.5, and all the experts had the same point of view, therefore, the indicator list was not changed at this stage.

According to the results of the Kolmogorov-Smirnov one sample test, there was no significant difference between Probe 2 and Probe 3, as shown in Table 4. And after three rounds of investigation, all the indicators were convergent. However, according to the results of Kruskal-Wallis one-way analysis, it showed that the experts' point of view from both the academia and the industry were significantly different, such as 1.6 Input and output, 3.2 Sequential access file, 4.3 Recursive function (subprogram), 5.1 One dimension array, 5.2 Two dimensions array, 5.3 Multi dimensions array, 7.2 Union, 7.3 Enumeration, 8.6 Polymorphism, 8.7 Virtual function and 8.8 Exception. Furthermore, the experts from academia thought that the indicators: 1.6 Input and output, 5.1 One dimension array and 5.2 Two dimensions array, to be more important than the experts from the industry did. However, the experts from industry thought the indicators: 3.2 Sequential access file, 4.3 Recursive function (subprogram), 5.3 Multi dimensions array, 7.2 Union, 7.3 Enumeration, 8.6 Polymorphism, 8.7 Virtual function, and 8.8 Exception, were more important than the experts from academia thought. Nevertheless, there were no different viewpoints in regard to the rest of the indicators.

AHP

In order to ensure that the data from the AHP are valid, Saaty suggested using the consistency index (CI) and the consistency ratio (CR) to test the consistency of the matrix [13]. When CI is zero, it presents high consistency between the pre and the post. However, complete sameness is difficult to achieve in real life. Therefore, Saaty said that CI $\leqq$ 0.1 is acceptable. The formula of the CI is CI = (λmax-n)/(n-1). CR: in the same layers of the matrix, the ratio of CI and RI is called the consistency ratio. Furthermore, the positive reciprocal matrix is produced from 1 to 9 scales of the measurement, and under the different orders, the random index (RI) value is produced. Consequently, CR equals CI times RI, and Saaty suggested that the CR value is equal or less than 0.1, then, the consistency is acceptable [13].

The results from the three probes of the modified Delphi analysis were used to investigate the differences between the capability indicators, in addition to using the weights for each dimension derived by using AHP to compare each hierarchy and item. In the end, 29 valid questionnaires were collected, and then Microsoft Excel and Expert Choice were used to process the data.

An internal consistency test was carried out, and found the CI value to be 0.006. Based on the research findings, the comparative weights of each dimension are shown in Table 5. The more important indicators in the programming design capabilities were found to be 2.3 Repeat flow controls, 2.1 Logic decision and 4.1 System.

Table 5. The weights and ranking generated by AHP.

Dimension	Dimensional weight	Indicator	Indicator Weight	Hierarchy Weight	Hierarchy Ranking
1. Basics of C++	0.105	1.1 Introduction to programming	0.106	0.011	38
		1.2 Introduction to C language	0.084	0.009	39
		1.3 Data types and identifier	0.194	0.020	25
		1.4 Constant and variable	0.183	0.019	27
		1.5 Operator and expression (modified)	0.228	0.024	17
		1.6 Input and output	0.204	0.021	20
2. Flow control	0.173	2.1 Logic decision	0.294	0.051	2
		2.2 Select flow control	0.218	0.038	6
		2.3 Repeat flow control	0.324	0.056	1
		2.4 Interrupt and break flow	0.163	0.028	14
3. File and application	0.060	3.1 Random access file	0.354	0.021	21
		3.2 Sequential access file	0.344	0.021	24
		3.3 Index access file	0.302	0.018	29
4. Subprogram and application	0.103	4.1 System function (subprogram)	0.473	0.049	3
		4.2 User-defined function (subprogram)	0.320	0.033	12
		4.3 Recursive function (subprogram)	0.206	0.021	22

Dimension	Dimensional weight	Indicator	Indicator Weight	Hierarchy Weight	Hierarchy Ranking
5. Array and application	0.203	5.1 One dimension array	0.211	0.043	4
		5.2 Two dimensions array	0.181	0.037	7
		5.3 Multi dimensions array	0.079	0.016	33
		5.4 String	0.192	0.039	5
		5.5 Searching	0.166	0.034	11
		5.6 Sorting	0.171	0.035	9
6. Linked list and application	0.079	6.1 Pointer	0.255	0.020	26
		6.2 linked list	0.155	0.012	34
		6.3 Searching	0.218	0.017	31
		6.4 Sorting	0.220	0.017	30
		6.5 Dynamic and static memory allocation (added)	0.151	0.012	35
7. Class and application	0.048	7.1 Structure	0.514	0.025	16
		7.2 Union	0.241	0.012	37
		7.3 Enumeration	0.245	0.012	36
8. Object and application	0.227	8.1 Introduction to class (added)	0.098	0.022	18
		8.2 Class and object(modified)	0.150	0.034	10
		8.3 Inheritance	0.159	0.036	8
		8.4 Encapsulation	0.134	0.030	13
		8.5 Dynamic linking	0.082	0.019	28
		8.6 Polymorphism (added)	0.116	0.026	15
		8.7 Virtual function (added)	0.072	0.016	32
		8.8 Exception (added)	0.093	0.021	23
		8.9 Pattern (added)	0.096	0.022	19

CONCLUSIONS

This study used a modified Delphi method and AHP to clarify the viewpoints of programming design capabilities from academic and industry experts, and found that the experts from these different sectors thought differently. Therefore, it appears the this issue is worth further and more comprehensive investigation.

In the finding of the modified Delphi analysis, the experts from the academia thought that the most important capability is One dimension array from Array and application. Secondary, Repeat flow control, Select flow control, Logic decision from Flow control, and Input and output from Basics of C++ are more important. Next in importance were String from Array and application, User-defined function (subprogram) from Subprogram and application, and Two dimensions array from Array and application. However, the experts from industry thought the most important items were Recursive function (subprogram), System function (subprogram), and User-defined function (subprogram) from Subprogram and application, followed by Polymorphism, Exception, Inheritance, and Class and object from Object and application.

In addition, String from Array and application is important to the industry experts as well. According to the findings above, there is a gap between the experts from academia and industry; therefore, when redesigning the curriculum, it is important to adjust the proportions of what the industry experts valued more, as it could decrease the sense of loss. Moreover, when the graduates enter the job market, this human resource is closer to what industry actually needs. It is an important issue when develop the programming design curriculum.

According to the results of AHP, Repeat flow control (weight is 0.056) is the most important capability in programming design, followed by Logic decision (weight is 0.051), System function (subprogram) (weight is 0.049), One dimension array (weight is 0.043), String (weight is 0.039), Select flow control (weight is 0.038), Two dimensions array (weight is 0.037) and Inheritance (weight is 0.036). The AHP results tend to be similar to the results from the academic experts, but still mainly focus on the capabilities of the dimensions in Array and application, and Subprogram and application.

ACKNOWLEDGEMENTS

The authors would like to thank the National Science Council of Taiwan for financially supporting this research under Contract Nos. <NSC 100-2511-S-241-007-MY3> and <NSC 102-2511-S-241-005-MY2>.

REFERENCES

1. Macias, M.E. and Guridi, E.D., Emulation of real processes to improve training in automation. *Inter. J. of Engng. Educ.*, 25, **2**, 358-364 (2009).

2. Verdú, M., Gómez-Aparicio, L. and Valiente-Banuet, A., Phylogenetic relatedness as a tool in restoration ecology: a meta-analysis. *Proc. Royal Society B: Biological Sciences*, rspb20112268 (2011).
3. Layman, L., Williams, L., Slaten, K., Berenson, S. and Vouk, M., Addressing diverse needs through a balance of agile and plan-driven software development methodologies in the core software engineering course. *Inter. J. of Engng. Educ.*, 24, **4**, 659-670 (2008).
4. Smith, S.S. and Saunders, K.P., Virtual reality for future workforce preparation. *Computer Applications in Engng. Educ.*, 17, **4**, 429-434 (2009).
5. Law, K.M.Y., Lee, V.C.S. and Yu, Y.T., Learning motivation in e-learning facilitated computer programming courses. *Computers & Educ.*, 55, **1**, 218-228 (2010).
6. Jenkins, T., On the difficulty of learning to program. *Proc. 3rd Annual Conf. of the LTSN Centre for Infor. and Computer Sciences*, 4, 53-58 (2002).
7. Hulls, C.C.W., Neale, A.J., Komalo, B.N., Petrov, V. and Brush, D.J., Interactive tutorial assistance for a first programming course. *IEEE Trans. on Educ.*, 48, **4**, 719-728 (2005).
8. Ala-Mutka, K., Uimonen, T. and Jarvinen, H.M., Supporting students in C++ programming courses with automatic program style assessment. *J. of Infor. Technol. Educ.*, 3, 245-262 (2004).
9. National Council on Technical and Vocational Education and Training. Assessment Guidelines (Version 6) (2005).
10. Finch, C.R. and Crunkilton, J.R., *Curriculum Development in Vocational and Technical Education: Planning, Content and Implementation.* (3rd Edn), Boston: Allyn and Bacon, Inc. (1989).
11. King, W.S. and Duan, L., Practical summer training in civil and construction engineering for cultivation of professional ability. *J. of Professional Issues in Engng. Educ. and Practice*, 136, **4**, 233-238 (2010).
12. Brill, J.M., Bishop, M.J. and Walker, A.E., The competencies and characteristics required of an effective project manager: a web-based Delphi study. *Etr&D-Educational Technol. Research and Develop.*, 54, **2**, 115-140 (2006).
13. Saaty, T.L., *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation.* New York, NY: McGraw-Hill International Book Co (1980).
14. Moore, C.M., *Group Techniques for Idea Building*. Sage Publications, Inc. (1987).
15. Faherty, V., Continuing social work education: results of a Dephil survey. *J. of Educ. for Social Work*, 15, **1**, 12-19 (1979).
16. Raskin, M.S.. The Delphi study in field instruction revisited: expert consensus on issues and research priorities. *J. of Social Work Educ.*, 30, **1**, 75-89 (1994).