

Computing Education in a Hybrid World

Computing Education in a Hybrid World

Laura Benvenuti



Laura Benvenuti

COMPUTING EDUCATION IN A HYBRID WORLD

Laura Benvenuti

ISBN: 978-94-6375-497-2

© Laura Benvenuti, Utrecht, NL, 2019, unless stated otherwise

Cover design and illustration by Martha Lauria Baca (www.lauria.nl)

Printed by Ridderprint BV (www.ridderprint.nl)

COMPUTING EDUCATION IN A HYBRID WORLD

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Open Universiteit
op gezag van de rector magnificus
prof. dr. Th. J. Bastiaens
ten overstaan van een door het
College voor promoties ingestelde commissie
in het openbaar te verdedigen

op vrijdag 20 september 2019 te Heerlen
om 13.30 uur precies

door

Laura Benvenuti
geboren op 30 maart 1962 te Etterbeek (België)

Promotores

Prof. dr. ir. J.M. Versendaal, Open Universiteit

Prof. dr. G.C. van der Veer, Open Universiteit

Leden beoordelingscommissie

Prof. dr. L.J.M. Nieuwenhuis, Universiteit Twente

Prof. dr. E. Barendsen, Open Universiteit

Prof. dr. M.C.J.D. van Eekelen, Open Universiteit

Dr. ing. J.P.P. Ravesteijn, Hogeschool Utrecht

For my father

TABLE OF CONTENTS

1	INTRODUCTION	1
1.1	Main research question	2
1.2	Readers' Road Map	2
1.3	Decomposed research questions	4
	PART I—WHAT IS TAUGHT AND WHY?	7
	Research questions RQ1, RQ2, RQ3, RQ6, RQ7	7
2	PRELIMINARY STUDY	9
2.1	Introduction	9
2.2	Computing curricula	11
2.3	Is computing a science?	12
2.4	Mathematical intermezzo	13
2.5	Verification and validation	14
2.6	Software Engineering	14
2.7	Two interpretations of <i>validation</i> in Software Engineering	15
2.8	Information Systems	16
2.9	One interpretation of <i>validation</i> in Information Systems	17
2.10	The nature of guidelines and recommendations	18

2.11 Discussion	19
2.12 So what?	20
3 HISTORICAL REVIEW AND EPISTEMOLOGICAL CONSIDERATIONS	23
3.1 Introduction	24
3.2 Computing curriculum guidelines	25
3.2.1 A theoretical approach	26
3.2.2 The software crisis	26
3.2.3 The Snowbird conferences in the 1980s	27
3.2.4 The ACM/IEEE curriculum reports	28
3.2.5 Hybrid computing curricula	29
3.3 Cultural styles in computing	29
3.4 Discussion: curricular trade-offs	31
3.4.1 Incorporation of the three cultural styles	31
3.4.2 Fostering discipline oriented thinking	32
3.4.3 The role of mathematics	33
3.5 Hybrid curricula: two cases	33
3.5.1 Liberal Arts and Computer Science	33
3.5.2 Front End Development	34
3.6 Conclusions	35
4 COMPUTING CURRICULA IN DUTCH UNIVERSITIES OF APPLIED SCIENCES	37
4.1 Introduction	37
4.2 Frameworks for curriculum recommendations	37
4.2.1 ACM/IEEE series	38
4.2.2 e-CF	40
4.2.3 Dutch Bachelor of ICT frameworks	42

4.2.3.1	Similarities and differences HBO-ICT 2009 & 2014	44
4.2.4	HBO Creative Technologies	45
4.3	Models for comparison	46
4.3.1	The Darmstadt model	47
4.3.1.1	Discussion	48
4.3.2	Van den Akker's curricular spider web	49
4.3.3	5 aspects	50
4.4	Curricular frameworks compared	51
4.4.1	Rationale	51
4.4.1.1	ACM/IEEE series	51
4.4.1.2	e-CF	52
4.4.1.3	HBO-ICT 2009	52
4.4.1.4	HBO-ICT 2014	52
4.4.1.5	HBO-Creative Technologies	53
4.4.2	Intentions: Learning Objectives	53
4.4.2.1	ACM/IEEE series	53
4.4.2.2	e-CF	53
4.4.2.3	HBO-ICT 2009	53
4.4.2.4	HBO-ICT 2014	53
4.4.2.5	HBO-Creative Technologies	54
4.4.3	Intentions: Competencies	54
4.4.3.1	ACM/IEEE series	54
4.4.3.2	e-CF	54
4.4.3.3	HBO-ICT 2009	55
4.4.3.4	HBO-ICT 2014	55
4.4.3.5	HBO-Creative Technologies	55
4.4.4	Intentions: educational standards	55
4.4.4.1	ACM/IEEE series	56
4.4.4.2	e-CF	56
4.4.4.3	HBO-ICT 2009	56
4.4.4.4	HBO-ICT 2014	56
4.4.4.5	HBO-Creative Technologies	56
4.4.5	Knowledge	57
4.4.5.1	ACM/IEEE series	57
4.4.5.2	e-CF	57
4.4.5.3	HBO-ICT 2009	58
4.4.5.4	HBO-ICT 2014	58
4.4.5.5	HBO-Creative Technologies	58

4.5 Discussion	61
4.5.1 The aims of undergraduate computing education	61
4.5.2 Hybrid curricula and new professional roles	63
4.6 Conclusions and recommendations	64
PART I - CONCLUSIONS	67
Research question RQ1	67
Recommendations	68
Research questions RQ2, RQ3	68
Recommendations	71
Research questions RQ6, RQ7	72
Recommendations:	72
PART II—HOW DO STUDENTS UNDERSTAND THE SUBJECT	75
Research questions RQ4, RQ6, RQ7	75
5 COGNITIVE ASPECTS OF SOFTWARE DEVELOPMENT	79
5.1 Introduction	79
5.2 Background	81
5.2.1 Mental models	81
5.2.2 Individual preferences	82
5.2.3 Assessing mental models: the teach-back protocol	84
5.3 Literature review	85
5.3.1 Cognitive aspects of (OO) programming	85
5.3.2 Cognitive aspects of user-database interaction	88
5.4 A need for empirical study	89
5.5 A first experiment: how do professionals understand their systems?	91
5.5.1 Questions	92
5.5.2 Scoring categories	93
5.5.3 Participants	93

5.5.4	Hypotheses	93
5.5.5	Preliminary Results	94
5.6	Preliminary conclusions	95
5.7	Acknowledgements	95
6	CONCEPTUALIZATIONS OF THE NOTION OF AN OBJECT	97
6.1	Introduction	97
6.2	Backgrounds	99
6.2.1	Mental models	99
6.2.2	Individual preferences	100
6.2.3	Assessing mental models: the teach-back protocol	100
6.3	Literature Review	101
6.3.1	Objects	101
6.3.2	Cognitive aspects of (OO) programming	102
6.3.3	Cognitive aspects of user-database interaction	103
6.4	A need for empirical study	103
6.5	Experiment design	104
6.5.1	Context	105
6.5.2	Questionnaire	105
6.5.3	Research questions and scoring categories	106
6.5.4	Reliability	108
6.5.5	Participants	109
6.5.6	Hypotheses	109
6.6	Results	110
6.6.1	Ha: amounts of different objects reported	110
6.6.2	Hb: mental models across the disciplines	112
6.6.3	Hc: problem solving preferences	114
6.7	Lessons learned	115
6.8	Conclusions	116

6.9 Our message for education	116
PART II – CONCLUSIONS	119
Research question RQ4	119
Research questions RQ6, RQ7	120
Recommendations	120
Further Research	120
PART III—CASE STUDIES IN A HYBRID CURRICULUM	123
Research questions RQ5, RQ6, RQ7	123
7 A CRAFTSMANSHIP-BASED APPROACH (1)	125
7.1 Introduction	125
7.2 New technology, new paradigm	125
7.2.1 Constructivist learning	126
7.2.2 Why 3D virtual worlds afford constructivist learning	127
7.2.3 Downsides of constructivist learning	128
7.2.4 Do 3D virtual worlds always support constructivist learning?	129
7.2.5 how to assess subjective impressions	129
7.2.6 Measuring experience	130
7.3 Observations in theory and practice	130
7.4 A course on 3D virtual worlds	132
7.4.1 Course structure	133
7.4.2 A virtual world as an educational tool	133
7.4.2.1 What does not work	134
7.5 Assessment	134
7.5.1 The survey	135
7.5.2 Results	136
7.6 Conclusions and future works	138

8	A CRAFTSMANSHIP-BASED APPROACH (2)	141
8.1	Introduction	141
8.2	A virtual village as a community of learners	141
8.3	Measuring experienced connectedness and learning	143
8.4	The asterix village	144
8.5	What's new, what's next?	145
8.6	Perceived connectedness and learning	148
8.7	Conclusions	150
PART III	- CONCLUSIONS	153
	Research question RQ5	153
	Research questions RQ6, RQ7	154
	Recommendations	155
	Further research	155
PART IV	—HCI IN A HYBRID CURRICULUM: RESEARCH IN ACTION	157
	Research question RQ7	157
9	HCI IN A HYBRID CURRICULUM: RESEARCH IN ACTION	159
9.1	Introduction	159
9.1.1	An excellent curriculum	160
9.1.2	Success factors	160
9.1.3	Corporate guidelines for the e-learning environment	161
9.1.4	Discussion	162
9.2	A workplace online	162
9.2.1	Actual experiment (Blackboard course site)	163

9.3 Design issues and solutions	164
9.3.1 Context and content both central	165
9.3.2 Encourage student participation in research	165
9.3.3 Share results	166
9.3.4 Practice what you preach and ask for feedback	166
9.3.5 Apply new concepts while adapting to organisational constraints	166
9.4 Preliminary evaluation	167
9.4.1 Conclusion	168
PART IV - CONCLUSIONS	169
Research question RQ7	169
Recommendations	169
Further research	170
10 OVERALL CONCLUSIONS AND RECOMMENDATIONS	171
10.1 Research question RQ1	173
10.1.1 Part I	173
10.1.2 Part III	174
10.1.3 Recommendations	175
10.2 Research questions RQ2, RQ3	176
10.2.1 Part I	176
10.2.2 Part III	178
10.2.3 recommendations	179
10.3 Research question RQ4	179
10.3.1 Part II	180
10.3.1.1 The differences we had expected	180
10.3.1.2 The differences we found	180
10.3.1.3 A possible explanation	181
10.3.1.4 Hypothesis and further research	183
10.3.2 Recommendations	183
10.4 Research question RQ5	183
10.4.1 Part III	184

10.4.2	Recommendations	184
10.5	Research questions RQ6, RQ7	184
10.5.1	Part I	185
10.5.2	Part II	185
10.5.3	Part III	186
10.5.4	Part IV	187
10.5.5	Recommendations	188
10.6	Which lessons can we learn that can be applied?	188
	REFERENCES	191
	SUMMARY	199
	SAMENVATTING	201
	CURRICULUM VITAE LAURA BENVENUTI	203

1 Introduction

Computing is an interdisciplinary field that can be approached from different points of view. Each point of view has its goals, aims and fundamental assumptions. In addition of this fragmentation, the field of computing is still rapidly evolving. Hybrid domains are emerging such as medical information systems. The domain of Creative Technologies, the interdisciplinary field of study combining computer technology, design and humanities, is one of the emerging disciplines in the computing related family. The importance of this new branch of computing for the Netherlands is uncontested. The related economic sector, Creative Industries, is one of the economic top sectors in the Netherlands.

Industry asks for skilled workforce. But skill requirements change rapidly in Information and Communication Technology (ICT)-related domains. This poses complex questions to designers of undergraduate computing curricula, especially of hybrid curricula. One possible solution is accommodation: a continuing re-adjustment, in order to match the most promising trends on the employment market. In the Netherlands, this is the leading strategy of the Universities of Applied Sciences. But in the long term, this strategy could put other aspects of education at risk, as the students' employability, or the development of the discipline. Do students acquire knowledge that is useful in the long term? Do computing professionals, educated in different areas of computing, understand each other's points of view? These questions are not new, as we will see in chapter 3.

In this thesis, we reflect on the education of computing professionals in general and of professionals in hybrid professions in particular. Our field of investigation is tertiary education. One of the challenges, for an inquiry on educational systems, is terminology. We will focus on the first stage after secondary education, the stage conferring a Bachelor's degree or Baccalaureate. The approach to education can be academic or applied, as in the European Universities of Applied Sciences. In accordance with the ACM/IEEE Curriculum Recommendations series, we will refer to this stage of education with the term "undergraduate education".

Our aim is to support designers of computing curricula, including designers of hybrid computing curricula. We will formulate recommendations for designing computing curricula in a way that (1) ensures graduates access to the labor market (2) allows them to keep up with their turbulent profession and (3) delimits these professions. This leads to the following Main Research Question.

Computing Education in a Hybrid World

1.1 MAIN RESEARCH QUESTION

The main research question in this thesis is:

Which lessons can be learned from past and present undergraduate computing education, which can be applied in the design of future undergraduate computing curricula and hybrid undergraduate computing curricula in particular?

1.2 READERS' ROAD MAP

Hybrid computing curricula draw our attention when we designed the computing content of a curriculum on Digital Communication. The units on Multimedia and Web Design were particularly challenging. At that time, the Multimedia sector used specific programming environments, designed to enable other software developers than “computer programmers” to implement their ideas. These tools were mainly used by artists. Something similar applies to Web languages. Web languages were developed as end user tools; they are user friendly and forgiving. Specific training seemed unnecessary to use these technologies. But the labor market develops over time, and new professional figures emerge as Web Developers, professional users of Web Technologies. Should students Web and Multimedia technologies be approached as future software developers – which would have meant including some general computing principles in the program, or was it sufficient to briefly explain the tools and set up a helpdesk? What does it mean to be a “professional user of software development tools”? Should all software developers be educated in the same way?

We explored these subjects with two papers. One described differences between the aims of the humanities on Multimedia education and those of computing (Benvenuti, L. & van der Veer, G.C., 2009). We pointed at different attitudes towards interpretation (is it acceptable that users freely interpret the output of software?) and at different expectations about the software's lifecycle. The other paper discussed the classification of computing: natural science or formal science. Although that paper raised interesting questions, it quickly became clear that the dichotomy scientific/formal did not cover the issue. You will find that paper in chapter 2.

At the same time, we observed the effects of computing education in Dutch Universities of Applied Sciences. We investigated how undergraduate students, educated in different computing programs, understand the abstract concept of “object”. We had the opportunity to observe a course in a

hybrid curriculum, where undergraduate students developed applications in a virtual world. The course had been enthusiastically welcomed by the students. We investigated their learning experience. We explored online instructional strategies in academic setting. Taking a hands-on approach, we designed a course on computing targeting a hybrid audience. Instead of simplifying, the overall picture became more and more diversified.

This changed when we found the work of Tedre and Apiola on three traditions of computing (Tedre & Apiola, 2013). Tedre and Apiola distinguish three traditions of computing, or cultural styles, fulfilling different roles in the development of the discipline: the theoretical, the scientific and the engineering tradition. Based on Tedre and Apiola's work, we consider these cultural styles as attempts to cope with the same fundamental problem.

Computing requires working with abstractions: models, structures, algorithms. Unlike mathematicians, who work with institutionalized abstract concepts, computing professionals often define the abstractions they work with (Dijkstra, *Programming as a discipline of mathematical nature*, 1973). This raises the question, how to sustain claims concerning these abstractions. The theoretical cultural style addresses this question formally, and describes abstract structures in an unambiguous way. The scientific cultural style addresses the question: do our models match with the world they describe? The engineering cultural style addresses the question, how to design and implement reliable systems.

Tedre and Apiola's considerations about three cultural styles of computing support reasoning about hybrid curricula, as we will see in chapter 3 (Part I). Their perspective turned out to be helpful in the discussion of the findings in our studies at the HU University of Applied Sciences Utrecht (Parts II and III). It enables understanding of the strengths and weaknesses of the course we designed for a hybrid audience (Part IV).

This thesis describes the details of that journey. Different reading strategies can be followed. To readers, who are mainly interested in the three traditions of computing, we suggest this reading path: start with chapter 2 (exploration), then read Part II (is there a problem?), chapter 3 (additional research), chapter 4 (computing curricula in Dutch Universities of Applied Sciences) and finally section 10.3.

To readers, who are mainly interested in hybrid curricula, we suggest to start with chapter 4 (computing curricula in Dutch Universities of Applied Sciences), then read Part III (how is this applied?), then chapter 3 (suggestions to type a hybrid curriculum), followed by Part IV (a suggestion for course design) and finally chapter 10 (Overall Conclusions and Recommendations).

Computing Education in a Hybrid World

1.3 DECOMPOSED RESEARCH QUESTIONS

The journey we are describing is one made of field research, case studies and reflection about computing education. After a general exploration of undergraduate computing education, we will narrow our inquiry to computing education at the Dutch Universities of Applied Sciences. We will focus on undergraduate computing curricula, and undergraduate hybrid curricula. We will discuss the outcomes of one experiment and a case study we conducted in Dutch Universities of Applied Sciences. We will comment on the way computing is taught today and has been taught in the past decades. We will make suggestions for improvement.

Finding a 'greatest common divisor' for this research has not been easy. An overarching theme would allow us to explicate which aspects of the overarching theme we have addressed, and which we have not.

Our decomposed research questions are:

- RQ1 What are possible approaches to computing and computing education?*
- RQ2 What are the aims of undergraduate computing education?*
- RQ3 What is the purpose of undergraduate computing curriculum recommendations series, i.e. of (RQ3a) international curriculum recommendations series and (RQ3b) curriculum recommendations series for Dutch Universities of Applied Sciences?*
- RQ4 Do students, who were educated in different programs, develop the same mental models for the abstract concepts they work with? I.e., are different approaches to computing interchangeable?*
- RQ5 How do students in a hybrid curriculum experience a craftsmanship-based approach?*
- RQ6 Which subject-specific strategies were recommended in the past?*
- RQ7 Which subject-specific strategies can we recommend?*

What our research questions do have in common is that they all concern knowledge and beliefs about education in computing related topics.

Magnusson et al. (Magnusson, Krajcik, & Borko, 1999) discuss this kind of knowledge, Pedagogical Content Knowledge, applied to science teaching. They conceptualize Pedagogical Content Knowledge of science teaching as consisting of five components:

- (1) orientation towards science teaching,
- (2) knowledge and beliefs about science curriculum,
- (3) knowledge and beliefs about students' understanding of specific science topics,
- (4) knowledge and beliefs about assessment in science and
- (5) knowledge and beliefs about instructional strategies for teaching science.

We will refer to these pedagogical components to delimit our inquiry. In this thesis, we investigate collective assumptions about undergraduate computing curricula, in particular about:

- (1) Orientations to computing education:
RQ1 What are possible approaches to computing and computing education?
- (2) Guidelines for computing curricula :
RQ2 What are the aims of undergraduate computing education?
RQ3 What is the purpose of undergraduate computing curriculum recommendations series, i.e. of (RQ3a) international curriculum recommendations series and (RQ3b) curriculum recommendations series for Dutch Universities of Applied Sciences?
- (3) Students' shared understanding of computing subjects,
RQ4 Do students, who were educated in different programs, develop the same mental models for the abstract concepts they work with? I.e., are different approaches to computing interchangeable?
RQ5 How do students in a hybrid curriculum experience a craftsmanship-based approach?
- (5) Instructional strategies specific to computing
RQ6 Which subject-specific strategies were recommended in the past?
RQ7 Which subject-specific strategies can we recommend?

The focus of our investigation has been on the first two components of Magnusson's PCK framework: approaches to computing and guidelines for undergraduate curricula. We have targeted specific issues of students' understanding. We have not investigated assessment. The discussion of instructional strategies is by no means complete; it consists of suggestions we have gathered during our investigations.

Part I—What Is Taught And Why?

RESEARCH QUESTIONS RQ1, RQ2, RQ3, RQ6, RQ7

RQ1 What are possible approaches to computing and computing education?

RQ2 What are the aims of undergraduate computing education?

RQ3 What is the purpose of undergraduate curriculum recommendations series, i.e. of (RQ3a) international guidelines for undergraduate computing curricula and (RQ3b) guidelines for computing curricula at Dutch Universities of Applied Sciences?

RQ6 Which subject-specific strategies were recommended in the past?

RQ7 Which subject-specific strategies can we recommend?

In this Part, we will reflect upon the discipline of computing and upon (Dutch) national and international curricular frameworks for undergraduate education.

Chapter 2 shows the first exploration of the question: *what are possible approaches to computing and computing education?* We start with the discussion of the classification of computing. We point at differences between validation according to Software Engineers and according to Information Systems specialists.

Chapter 3 expands the discussion to the whole discipline of computing, including hybrid disciplines. We characterize computing as a field that can be approached from three points of view: theoretical, scientific and engineering. We illustrate how this description can be adopted to better understand hybrid curricula.

Chapter 4 treats differences between (Dutch) national and international curricular frameworks for undergraduate computing education. It includes examples of the positioning of hybrid curricula.

2 Preliminary Study¹

ABSTRACT

Computing, or informatics as we call it in Europe, covers many areas. In this paper we will discuss an important difference between two of these areas: software engineering and information systems. Epistemology, the study of the question: "What grounds can we justifiably have for believing the truth of assertions about reality?", is complex in informatics. This question has different answers, depending on the area we investigate. Curricula in informatics do not discuss this difference explicitly. In our opinion, they should.

KEYWORDS

Computer Science Education, Information Science Education, Intellectual Discipline, Cultural Differences

ACM CLASSIFICATION KEYWORDS

K.3.2 [Computer and Information Science Education]: curriculum

2.1 INTRODUCTION

Computer applications increasingly determine how we live. Our choices depend on the information we retrieve using (mobile) apps, the same applies to the services we as citizens interact with. These applications are written by computing practitioners. Some of them are trained on the job, others have an academic or a professional degree.

We will discuss one aspect of the education of computing practitioners. When we write "computing" we also mean "informatics", unless it is perfectly clear that we intend to attribute different meanings to these two words. We choose "computing" to remain consistent with the terminology of the ACM-IEEE curriculum recommendation series. We have considered

¹ This work was originally published as: "Sciences, Computing, informatics: who is the keeper of the Real Faith?", In: G. van der Veer, P. Sloep, M. van Eekelen (eds) Proceedings of Computer Science Education Research 2011, April 7 and 8, Heerlen, the Netherlands. © 2011 Open Universiteit, Heerlen, NL. ISBN: 9789035819870

Computing Education in a Hybrid World

using “computer science” to indicate the discipline, but rejected the option because the curriculum recommendation series use that term to indicate one of the computing disciplines (see section 2.2). Incidentally, we will use “computer science”, but only while quoting other authors. Finally, we will use “informatics” in the discussion of European curricula in section 2.2 because that is how the discipline is called in Europe.

In this paper we express our concern about the education of computing practitioners. The discipline is young, and discussion about its focus and boundaries is taking place (Cassel, 2007). At the same time, the industry demands skilled professionals. How curricula in higher education should approach computing is a relevant question. If computing is seen as a competence, it is appropriate to focus on guidelines and recommendations - and how to apply them. But computing is an intellectual discipline too (Wing, 2006) (Lewis, Jackson, & Waite, 2010), where creativity is accompanied by reasoning. From that point of view, it is appropriate to focus on the discussion of choices.

This discussion is not new. Software Engineering (SE), is aware of the problems encountered by practitioners. Making choices is an important issue in the undergraduate curriculum in SE (ACM / IEEE, 2004, p. 40), Curriculum Guideline 8 (section “exercising critical judgment”) that is covered by offering a variety of methods and their backgrounds (Guideline 7 (section “computing”)).

In our opinion, confining the discussion of the philosophical underpinning of methods in the academic setting is not enough. Future programmers should be able to motivate their choices in a setting we do not know yet. They should have tools to evaluate the methods they are acquainted with. They also should develop an intellectual attitude towards the profession of programmer.

Lewis, Jackson and Waite (Lewis, Jackson, & Waite, 2010) looked at side effects of higher education in our discipline. They investigated student attitudes early and late in an undergraduate Computer Science (CS) curriculum. One of the statements they proposed to 1st semester students, 2nd semester students and senior level students was: “When I solve a computer science problem, I explicitly think about which computer science ideas apply to the problem”. The staff would have wanted the students to endorse this statement, and many of them did, but student endorsement declined from 72% for 1st semester students to 44% for senior level students. Lewis, Jackson and Waite’s conclusion was that the CS curriculum might fall short in helping students develop the perspective that CS is an intellectual discipline.

Undergraduate computing curricula should emphasize the intellectual aspects of the disciplines, besides the competences. After all, today's students will be designing tomorrow's world. Discussion is necessary.

In section 2.2, we will look at undergraduate computing curricula in European higher education; in section 2.3 and 2.4 we point at a discrepancy between the administrative classification of computing and the historical roots of the discipline. In the sections 2.6 and 2.7 we discuss the nature of guidelines in software engineering; in 2.8 and 2.9 we look at the same aspect of another computing discipline, information systems. In section 2.10 we compare the approaches of these two disciplines. In the sections 2.11 and 2.12 we draw conclusions for the professional practice as well as for the academic curricula.

2.2 COMPUTING CURRICULA

In Europe, there is little agreement on the organization of higher education in computing or informatics. Harmonization is taking place (EHEA, sd) but it is far from being accomplished. The United Kingdom has a system that resembles the US standard (Cowling, 2006). France has university education and top graduate schools or "Grandes Écoles" (Commission des Titres d'Ingénieur) awarding prestigious Master's degrees. Flanders (NVAO, sd), and the Netherlands (NVAO, sd) have a dual system with institutes for Higher Professional Education (Hogescholen) and Universities. Hogescholen generally confer Bachelor's degrees; in Flanders these degrees can be "academic" (Vlaamse overheid, sd) while in the Netherlands they never are (NVAO, 2008). In Italy there is one system of academic degrees (Ministero dell'Istruzione, dell'Università e della Ricerca, 2000) that includes Engineering and Architecture disciplines. We will focus here on the first academic degree in computing or informatics, the (academic) Bachelor's degree.

The name of the discipline also reflects different views. In Europe, we use "informatics" as an umbrella term (Informatics Europe, sd), except for the U.K. where "computing" is preferred, in analogy with the U.S.A. (Cowling, 2006). The first name suggests an emphasis on automated information processing, the second on the computerization of calculus.

Surprisingly, there is agreement on the classification of the discipline. The conferred academic degree in Europe, including the UK, is B.Sc. From a legal point of view, informatics is one of the Natural Sciences. In the Netherlands, it falls under the area "Nature", with Mathematics and the Natural Sciences, which is often bundled with Technology into the area of

Computing Education in a Hybrid World

“ β -wetenschappen” (β sciences) (Larsen & Lubbe, 2008), or “Exacte Wetenschappen” (exact sciences) (NWO, 2011). In Italy, the Informatics curricula are classified as “Scientific” and “Engineering” (Ministero dell'Istruzione, dell'Universita' e della Ricerca, 2000), in France (Comité de Suivie de la Licence, 2010) Informatics is listed in the domain “Sciences, Technologies et Santé” (Sciences, technology and health). In the UK, Computing is considered part of Science, Technology, Engineering and Mathematics (STEM) [26]. There are rare exceptions, for example the University of Groningen (NL) offers a full Information Systems-curriculum in the Faculty of Arts (Rijks Universiteit Groningen, Faculty of Arts, sd). It confers a BA degree.

The ACM-IEEE Joint Task Force for Computing Curricula has defined a classification for academic curricula in the USA and in the UK (ACM / IEEE, 2005), as an overview for the Computing Curricula series with guidelines and standards. There is no agreement (yet) on the content of academic curricula in informatics in Europe, but many national institutions for curriculum evaluation are inspired by the Computing Curricula recommendations, such as the Italian “Bollino GRIN” (Cortesi & Nardelli, 2007) and the last Dutch report on the quality of Bachelor’s degrees in Informatics (QANU, 2007). For this reason, we will follow the Task Force in the terminology we will use, in particular in its definition of undergraduate computing curricula, i.e. of the bachelor’s degrees in computer engineering, computer science, information systems, information technology and software engineering.

2.3 IS COMPUTING A SCIENCE?

The Task Force for Computing Curricula defines “computing” as “any goal-oriented activity requiring, benefitting from or creating computers” (ACM / IEEE, 2005), § 2.1). This definition is followed by the observation, in the same paragraph, that “an information system specialist will view computing somewhat differently from a software engineer”, each computing area having its focus and perspective. The definition of robust methods for creating reliable artefacts is at the core of the SE agenda, whereas the information system (IS) perspective emphasizes generating, processing and distributing information using computer technology. These differences are seen as gradual; the common interest is computing.

European administrations too tend to consider computing as one discipline, one of the natural sciences. That is not obvious at all: in computing, unlike in the natural sciences, the object of investigation is

artificial. Computing has that in common with engineering, but unlike in the case of engineering, the proof of soundness of computing theories is not necessarily situated in the outside world. Computing leans heavily on mathematics and formal logic.

The question “is computing a science” is a recurrent one. In 2007, P. Denning wrote in the Communications of the ACM “Computing is a Natural Science” (Denning P. , Computing is a Natural Science, 2007). In his opinion, computing has never been just a science of the artificial, but is an activity revealing deep structures of various natural processes. Denning also remarked that the acceptance of computing as a science is a recent development, that has taken place in less than a generation. Three years later, G. Génova reconsiders the scientific status of computer science (Genova, 2010) and argues for more speculative research in the discipline. The pendulum has swung too far in the direction of experimentalism. Experimentation and speculation should go hand in hand in all sciences but in computer science in particular. This discipline owes too much to theoretical research to focus principally on experimentation, says Génova.

2.4 MATHEMATICAL INTERMEZZO

Computing theories have their origin in D. Hilbert’s formalist school, one of the answers to the 19th Century’s crisis in the philosophy of mathematics. The German mathematician and philosopher David Hilbert wanted to provide for the foundation of number theory – and mathematics – in an axiomatic way, through formal logic. If number theory turned out to be consistent, then there had to be some sort of truth in it. Wegner and Golding (Wegner & Goldin, Principles of problem solving, 2006) call this a rationalist point of view: Hilbert considered mathematical knowledge independent of sense experience.

We owe the theory of computability to Hilbert and his school (Lolli, 2006) in the early 20th century at the University of Göttingen, that included Ernst Zermelo and John von Neumann (Hilbert, sd). But Hilbert’s program failed. We know now that number theory cannot be grounded in formal logic. Nor can mathematics, as an extension of number theory. We still do rely on mathematics and number theory, but the reason why is less clear.

In sections 2.6 to 2.9 we will explore the role of mathematics in two computing disciplines: software engineering and information systems.

Computing Education in a Hybrid World

2.5 VERIFICATION AND VALIDATION

One of the fundamental issues for programmers is: how do we know that the machine will always do what we want it to do? The question has two aspects: 1. Did we build the right program (Validation) and 2. Did we build it right (Verifications and Validation of software, sd)

According to the General Principles of Software validation of the FDA (Food and Drug Administration, 2002) “Software verification provides objective evidence that the design outputs of a particular phase of the software development life cycle meet all of the specified requirements for that phase.” (§ 3.1.2). In large and complex projects, software is specified formally. In these cases, verification concerns the correspondence between code and formal requirements and can be supported by formal proofs.

The FDA considers software validation to be “confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled.” (Food and Drug Administration, 2002), § 3.1.2). Validation concerns the correspondence between the product and what is needed in the real world. We will focus on this aspect of computing: are we making artefacts that are “valid”?

The story is becoming less “exact” at this point. In fact, there are different opinions about what should be considered “valid”: does the adjective apply to the product, to the process, to the stakeholders’ requests, or to the stakeholders’ needs?

In real life, the concept of “users” can be considered in a narrow way as “anybody who actively uses the system” or in a broad way “anybody who is client of the services provided by the system”. The second variant of the concept includes stakeholders who never touch the system but who “benefit from”, or are “victim of” others using the system. Validity, in this case, concerns not only stakeholder specifications, but also stakeholder understanding and acceptance. We will coin this broad concept “stakeholder validity”.

2.6 SOFTWARE ENGINEERING

The first area of computing we will investigate is software engineering. The most recent version of the “IEEE/ACM Software Engineering Curriculum Recommendations” specifies the nature of the discipline: “Software engineering thus is different in character from other engineering disciplines, due to both the intangible nature of software and to the discrete

nature of software operations. It seeks to integrate the principles of mathematics and computer science with the engineering practices developed to produce tangible, physical artifacts.” (ACM / IEEE, 2004, p. 6)

Software engineering uses mathematics; but why? Formal methods are used to describe software in order to control the process of software construction: “The mathematical and engineering fundamentals of software engineering provide theoretical and scientific underpinning for the construction of software products with desired attributes. These fundamentals support describing software engineering products in a precise manner. They provide the mathematical foundations to model and facilitate reasoning about these products and their interrelations, as well as form the basis for a predictable design process”. (ACM / IEEE, 2004, p. 23)

Software, at least critical software, is described in a formal way first and translated into code afterwards. The question “does code meet its formal specification?” (verification) is answered by formal, mathematical proof. The other crucial question (does the code do what it was designed for?) can be answered in different ways: we can validate the correspondence between formal specifications and the stakeholders’ requirements, we can validate the correspondence between formal specifications and his understanding and acceptance (stakeholder validity) and we can validate the software in the same way we validate other artifacts.

2.7 TWO INTERPRETATIONS OF VALIDATION IN SOFTWARE ENGINEERING

The IEEE/ACM Software Engineering Curriculum Recommendations is ambiguous at this point: “Requirements represent the real-world needs of users, customers, and other stakeholders affected by the system. The construction of requirements includes an analysis of the feasibility of the desired system, elicitation and analysis of stakeholders’ needs, the creation of a precise description of what the system should and should not do along with any constraints on its operation and implementation, and the validation of this description or specification by the stakeholders” (ACM / IEEE, 2004, p. 25)

We can read “precise descriptions” in two ways: as “formal specifications” or as “description by a finite number of objective, measurable criteria”.

In the first interpretation, software is meant to fulfill a formal description. In this case, the software engineer is charged with the translation between the stakeholders’ requirements, understanding and/or

Computing Education in a Hybrid World

acceptance and the formal specifications of the product. In order to increase the confidence in the outcome of this process, future software engineers are provided with a solid background in mathematics. Mathematics is also used to design software and to prove its correctness. We trust this approach because we trust mathematics. This is the research area Génova wants to safeguard.

In the second interpretation, software is considered as an artefact, an answer to well defined and measurable needs. The process of translating those needs into formal specifications and from formal specifications to software is evaluated as a whole. In this interpretation of “precise description”, we trust the validation process because we trust our observations, as we do in the natural sciences.

2.8 INFORMATION SYSTEMS

In the ACM/AIS Curriculum Guidelines for Undergraduate Degree Programs in Information Systems we read: “Information Systems as a field of academic studies encompasses (...) acquisition, deployment, management and strategy for information technology resources and services (...) and packaged system acquisition (...) for use in organizational processes.(...) The systems that deliver information and communication services in an organization combine both technical components and human operators and users. They capture, store, process, and communicate data, information, and knowledge.” (ACM / AIS, 2010, p. 13).

The Foundations of Information Systems course “is designed to introduce students to contemporary information systems and demonstrate how these systems are used throughout global organizations. The focus of this course will be on the key components of information systems - people, software, hardware, data, and communication technologies, and how these components can be integrated and managed to create competitive advantage.” (ACM / AIS, 2010, p. 36).

The operationalization is taught in the Data and Information Management course, which “provides the students with an introduction to the core concepts in data and information management. It is centered around the core skills of identifying organizational information requirements, modelling them using conceptual data modelling techniques, converting the conceptual data models into relational data models and verifying its structural characteristics with normalization techniques, and implementing and utilizing a relational database using an industrial-strength database management system.” (ACM / AIS, 2010, p. 40).

The requirements are translated into relational data models and implemented with a relational database. The relational database management system is supposed to be provided by the industry fully conforming to the relational model.

Relational data models are formal representations of existing structures: libraries, population registers etc.. They are implemented into relational databases, which are filled with data representing the actual situation in the real world, or at least of the portion of the real world represented by the system.

This approach is based on the assumption that formal reasoning preserves truth. Let us consider the case of the library. The librarian wants to answer questions about his library, by querying the library database. The queries are written by a database engineer. The engineer assumes that the digital representation describes the real library correctly. He formulates each query in terms of entities in the model and convinces himself by formal reasoning that the queries correspond with the questions he wants to answer. If the questions are correct and the digital representation of the library matches the library, the answers to the queries should be true.

We trust the outcome of database queries because we trust formal reasoning.

2.9 ONE INTERPRETATION OF *VALIDATION* IN INFORMATION SYSTEMS

In the information systems discipline, the only possibility to describe software requirements is the formal one. The issue of the correctness of database management software is delegated to the software houses, which should implement the relational model, a mathematical model. Requirements for information systems are always formal; all the communication between professionals concerning information systems is supported by mathematics. This applies both to communication between practitioners and to communication between practitioners and their software houses. The other variant does not apply here.

One could reply that, even if the specifications of information systems are formal, their content concerns the real world. Can we consider these systems as artefacts, is it possible to skip the verification/validation sequence and match the input/output with real world situations? There are two reasons to answer “not always” to this question.

The first one concerns the system’s dimension. It is always possible to match the real world situation with a positive answer to a database query,

Computing Education in a Hybrid World

but this does not apply to the negative answer. I can look up a title in a library. The answer “You will find the book at location XX” can easily be checked. The answer “This book is not available” can only be confirmed by checking every single book in the library. This is possible in a small library, but most present-day libraries are too large and, moreover, they are distributed systems; the books are stored in different locations. Confirmation of the negative answer is seldom feasible in practice.

The second reason concerns the human factor in information systems and its dynamics. What is likely to happen in a very large library, where it is impossible for the operator to overview the situation, is that the operator will trust the system’s negative answer and will discard the book when he happens to find it. The system’s state will match the real world situation again, but the reason of the match is far from being scientific. We cannot validate this software by matching the system’s behavior with the real world because, unlike in the natural sciences, the “world” described by information systems is made to comply to the model.

2.10 THE NATURE OF GUIDELINES AND RECOMMENDATIONS

We have discussed the validation of software so far. The question we focus on here is not “why do we think our software is correct” but “why do we trust our guidelines and recommendations?”

Epistemology is the branch of philosophy studying the nature and limits of knowledge. One of the discussions in epistemology concerns rationalism versus empiricism. Rationalism claims that pure reason can be a source of knowledge. The opposite position is held by empiricism, claiming that all knowledge has its origin in sense experience (Formal Sciences, sd). Sciences, and natural sciences in particular, are essentially empiricist. Scientific knowledge is refined by application of the scientific method that is based on empirical and measurable evidence. The status of mathematics is the subject of debates. (Formal Sciences, sd) (Putnam, 1975).

Validation of software is so complex because it concerns both the software in question and the discipline itself. If in software engineering, software performs unwanted behavior despite of having been tested meticulously, and the reason why cannot be found, the discipline reconsiders its guidelines. Of the two interpretations of “validation” (the correspondence of the requirements and the formal specifications versus validation by empirical evidence), the empirical variant is the one that

counts here. The discipline practices the scientific method. Epistemology is principally empiricist in software engineering.

Validation by empirical evidence is undeniable if the software is meant to produce results that are observable to everybody, if it controls processes in the physical world: software that commands the Mars Lander, or the Dutch Delta Works etc. Peter Denning's claim that computing reveals deep structures of nature matches with this approach to computing.

Validation of information systems occurs principally by comparing test results with their abstract counterpart. This process always entails a translation from the system's output to the abstract model. There is some objectiveness in this translation: professionals agree on how it should take place and agree on the definition of "malfunction of the system". There is less agreement on the definition of "malfunction" among the uninitiated ones. In real-life application of information systems, errors can remain invisible. Information systems relies principally on formal methods to validate both its software and its guidelines. The underlying assumption seems to be: if the model is consistent, then there has to be some truth in it. Epistemology appears to be principally rationalist here.

2.11 DISCUSSION

Apparently, the scientific interpretation does not fit the discipline of computing or informatics as a whole. In particular, it does not fit the construction of software that implements abstract models. Here, there is some room for different views on the status of theories, guidelines and recommendations. Different branches of computing seem to have different reasons to trust their guidelines. Who is the Keeper of the Real Faith? Our conclusion is: the question does not fit the discipline anymore

The discipline has expanded. There are areas where the ultimate test lies in the real world and in the perceived functioning of software. In other areas the ultimate test is formal. Practitioners (and their stakeholders) should be made aware of this issue.

Information processing is a key issue in our discipline. But there is no decisive answer to the question what information is. One of the ideas on this subject was translated into a formal model, the relational model, and laid the foundation for extremely successful technology. Working with that technology requires a formal approach to the discipline.

We doubt the rationalist assumption "if the model is consistent, there has to be some truth in it". In particular, we reject the application of the notion of "truth" or "correctness" to software producing results that cannot

Computing Education in a Hybrid World

be evaluated by the end users. What is the opposite of “correctness” in this case: “error” or “failure”? The unskilled user will only detect the latter.

Concerning the “validation” of software implementing an abstract model, it appears adequate to add criteria to the notion of “correctness”, criteria expressing stakeholder validity. Among them, we mention “applicability”, “usability”, “efficiency” or “effectiveness”. On which criteria to focus is a choice that depends on the context, a choice that precludes other choices. Therefore, it should be made explicitly.

Concerning the role of formal models in the computing practice, there seems to be more room for discussion on this point than most undergraduate curricula suggest.

2.12 SO WHAT?

We question the classification of computing among technology and the natural sciences because it blurs the objectives of the academic location of a discipline. An impartial discussion about the status of computing theories is unlikely to happen inside a faculty that has its right to exist in one of the possible outcomes of that discussion.

In our view, an impartial discussion might well end in the division of the discipline in “computing” (in accordance with the natural sciences), “engineering” and “informatics” (matching other criteria, depending on the context).

With regard to professional ethics, we witness that the classification of our discipline in the domain of the natural sciences entails an attitude of taking the benefits of technology too much for granted. We are afraid that the library will fit the designed system in the end. We want the system to fit the library instead. To facilitate this, future practitioners should be educated in discussing the criteria for application of the technology they use. They should be aware of the downsides of technologies, besides their benefits.

As regards the educational programs, we recommend academic curricula in all the computing disciplines to pay more attention to this issue. Today, students get acquainted with different branches of computing, without discussing the differences. Academic curricula cover different application areas with the corresponding methodologies, without a discussion of the reason why different contexts request different methods.

We recommend discussing the backgrounds of theories explicitly. Each field of application has its own needs and accents: aerospace application programmers follow different guidelines than Web artists. Awareness of the

Preliminary Study

backgrounds of theories helps practitioners to determine if the corresponding guidelines can be applied and how to do it properly.

3 Historical Review And Epistemological Considerations²

ABSTRACT

Computing is an interdisciplinary field that can be approached from different points of view. Each point of view has its goals, aims and fundamental assumptions. This makes computing a complex discipline. Moreover, new computing disciplines appear regularly. With the trend that ICT-professionals should have non-ICT competences as well, and non-ICT-professionals should have ICT-competences, new computing curricula are often hybrid in nature. As a hybrid computing curriculum cannot cover the full range of computing, it is interesting to investigate the ‘computing part’ of such curricula.

Our analysis framework consists of three elements: the curricular components ‘goals and objectives’ and ‘instructional strategies’, and the underlying epistemological view on the discipline (‘cultural styles’). Taking a historical perspective, we describe the origins of the ACM/IEEE Curriculum Recommendation series. We discuss the three main cultural styles of computing: theoretical, scientific and engineering.

Observing that in a curriculum the above elements should be aligned, we present three trade-offs for the case of hybrid computing curricula. We apply our results to two concrete examples, Liberal Arts and Computer Science and Front End Development. Based on our investigation, we formulate recommendations for designers of hybrid computing curricula. We recommend, for example, discussing disciplinary boundaries and resulting trade-offs explicitly while designing and documenting curricula.

CCS Concepts

• Social and professional topics ~ Computing education programs •
Social and professional topics ~ History of computing

Keywords

²This work was originally published as: Benvenuti, L., Barendsen, E., van der Veer, G. C., & Versendaal, J. (2018) Understanding computing in a hybrid world, on the undergraduate curriculum Front End Development. *SIGCSE'18, February 21-24, 2018, Baltimore, MD, USA* © 2018 ACM, NY, USA. ISBN 978-1-4503-5103-4, doi: 10.1145/3159450.3159532

Computing Education in a Hybrid World

Interdisciplinary programs (CS+X); Hybrid undergraduate curricula;
Cultural styles in computing; Front End Development

3.1 INTRODUCTION

New computing curricula often are hybrid in nature. Since the introduction of the World Wide Web, new areas, commonly referred to as Creative Technologies, have gained importance. Educational programs appear, meant to foster these developments by stimulating crossovers between computing and non-technological sectors as fashion, health etc. The corresponding curricula only partially concern computing, since a considerable part is devoted to sector specific topics. Designing curricula for hybrid contexts poses interesting questions. Which part of computing is relevant for hybrid programs, in how far should we consider graduates of these curricula as computing professionals? The answer to these questions is controversial.

Recent research on the Dutch labor market (Dialogic & Matchcare, 2016) observes that boundaries between Information and Communication Technology (ICT) and other sectors are fading away. It refers to the European e-Competence Framework (eCF) (European Committee for Standardisation, 2014) that divides specific ICT competences in primary ICT skills, as software development, and secondary ICT skills supporting primary skills, like information security strategy development of user support. Specific ICT competences are increasingly requested in other professions than the computing professions listed in eCF. If professional profiles were classified by the required ICT competences, argues the Dutch report, the computing professions would double. The report predicts a shortage of professionals having primary ICT skills in the immediate future, in particular a shortage of software developers. It also warns against the growth of hybrid undergraduate computing curricula, where 'hybrid' is defined as: "programs that primarily concern other areas than computing". According to the Dutch researchers, hybrid curricula focus too much on secondary ICT skills to comply with the industry's demand.

We question the researchers' conclusions about the education of software developers. But we do acknowledge that designing computing content for hybrid setting is a challenge. This paper strives at understanding computing, to scaffold choices, necessary to design hybrid curricula. Which part of computing is relevant in hybrid contexts, and why?

In our pedagogical analysis we consider two central curricular themes (cf. (van den Akker, Curriculum perspectives, an introduction, 2004)), namely

Historical Review And Epistemological Considerations

Goals and objectives of computing curricula and Instructional strategies specific for teaching computing. The underlying cultural or epistemological view on the discipline (Tedre & Apiola, 2013) is a third theme, which we refer to as Cultural styles. In a balanced curriculum, the above three themes are aligned, that is, goals and instructional strategies fit together and are consistent with an underlying cultural style.

We will conduct our investigation from a historical perspective. In Section 3.2 we will sketch the preamble for the joint ACM/IEEE series of curriculum guidelines, to better understand the international guidelines for the undergraduate computing curricula, in particular with respect to goals and objectives of computing curricula.

In Section 3.3, we will further examine the interdisciplinarity of computing. Besides a comparison of the different cultural styles in computing, we will discuss the implications for computing education, especially concerning instructional strategies.

We explore the challenges for hybrid curricula in terms of trade-offs between the three themes in Section 3.4. In Section 3.5 we apply our analysis to two particular hybrid curricula, Liberal Arts and Computer Science and Front End Development. We conclude with a summary and recommendations in Section 3.6.

3.2 COMPUTING CURRICULUM GUIDELINES

North America has a tradition of dialogue between professionals covering different roles in the computing community: scholars, professionals, engineers, and theoreticians. This is remarkable; in the Netherlands, the computing community was shaped by debates and is still divided today (Dael, 2001). In this section, we will explore the preamble to the Joint Task Forces on Curriculum Recommendation, that were initiated in North America but operate worldwide today.

The usage of programmable computing machines for practical purposes took off with the Second World War in the United States, Germany and Great Britain. But the European computers did not survive the war. The German Z3 was destroyed in an allied bombardment of Berlin in 1943; the British Colossi were considered highly classified and were dismantled for that reason. Things went differently in the U.S.A. The ENIAC was announced to the press in 1946. In the summer of that year, the Pentagon organized lectures about the construction of digital electronic computers, and invited computing professionals from the United States and Great Britain (Moore

Computing Education in a Hybrid World

School lectures, sd). The dissemination of the ideas underlying computing construction starts here.

Societies for computing professionals were quickly established: one of the precursors of IEEE (the Subcommittee on Large-Scale Computing of the American Institute of Electrical Engineers) in 1946, followed by ACM in 1947. American organizations of computing professionals have collaborated since the early days (Joint Computer Conferences, sd) by sponsoring joint conferences.

Tedre (Tedre, 2007) draws a detailed picture of the debates that have shaped the discipline in the second half of the 20th Century. Until the 1960s, mathematicians collaborated with engineers to realize computing machines and languages to program them. These activities resulted in artifacts that have an impact on the world, an impact that sometimes is measurable, but not always. Should computing be considered an art or a science? Should practical topics as programming be taught in academia; is computing an academic discipline at all?

3.2.1 A THEORETICAL APPROACH

The first edition of the ACM curriculum recommendations, published in 1968, took a stance in this debate. Computer Science (CS) had to be considered an academic discipline studying information structures and processes (Tedre, 2007). Its core issue was handling abstraction. Mathematics provided for theoretical foundation of the discipline. In these days, computer scientists were accommodated in Departments of Mathematics or in Departments of Electronic Engineering. Training for the positions of programmers could be supplied by technology programs, vocational education or junior colleges.

3.2.2 THE SOFTWARE CRISIS

The theoretical approach appeared not to be able to solve all the problems of the booming discipline. In the late 1960s, software complexity increasingly had become problematic. Complexity did not only arise from the parts of a system – object of study in the Computer Science (CS) departments - but from the connections between these parts (Tedre, 2007). This software crisis uncovered a lack of methodology for understanding and controlling software systems that were too big for single programmers. Software Engineering (SE) emerged, a discipline that considers software development as the effort of a team whose members might have contrasting intentions.

Historical Review And Epistemological Considerations

The update of the ACM Curriculum in 1978 brought programming back in the academic curriculum. With the software crisis, the focus of computing had shifted from a discipline studying information structures into an application-centered discipline (Tedre, 2007). Computing had gained independence from mathematics and engineering, but a clear definition of the discipline had been lost.

Curriculum '78 has been criticized for too strongly identifying computing with programming. In their objection to that view, Ralston and Shaw (Ralston & Shaw, Curriculum '78 is Computer Science really that unmathematical?, 1980) stated that mathematics lays a foundation under both science and engineering, and that understanding of any of these disciplines is impossible without understanding mathematics. Also, the mathematic foundations of computing are stable, where specific skills rapidly become obsolete. Giving students a background in mathematics would protect them from this obsolescence, pleaded Ralston and Shaw.

3.2.3 THE SNOWBIRD CONFERENCES IN THE 1980S

Starting with 1972, the Heads of CS Departments offering PhD in the USA and Canada held biennial meetings in Snowbird, Utah. Members of government and representatives from the computer industry attended these meetings, which evolved towards meetings of the North American computing community.

The 1980 Snowbird Report "A discipline in crisis" (Denning, et al., 1981) depicts a worrisome situation at Computer Science departments. The computing field was growing explosively. Universities were overburdened by the growth of undergraduate enrollments. There was no time for supervision of graduate students. The best students were hired by industry, which caused a shortage of PhDs. The situation threatened the ability to conduct basic research in Computer Science.

As an answer to the problems described in "A discipline in crisis", dialogues were improved between academia, the industry, government agencies and professional organizations (Yau, et al., 1983). The 1984 report (Future Issues in Computer Science (Tartar, et al., 1984) addresses the question of the lack of agreement about the accreditation of undergraduate programs. ACM and IEEE had started investigating the desirability of cooperating in the accreditation process at the 1982 Snowbird conference (Yau, et al., 1983). By 1984, the professional organizations had created a Computer Science Accreditation Board (CSAB); the volunteers necessary to administrate the CSAB were provided by academia.

Computing Education in a Hybrid World

One of the aims of CSAB was: providing a more common view of a core description, at least for the undergraduate curriculum. ACM and IEEE formed a joint task force to describe the intellectual substance of the computing field in 1985 (Denning, et al., 1989). A joint task force for the definition of the undergraduate curricula was formed in 1988 (Reilly, 2004).

3.2.4 THE ACM/IEEE CURRICULUM REPORTS

The Task Force, chaired by P. Denning, presented its final report “Computing as a Discipline” in 1989. The Task Force uses the phrase discipline of computing as an umbrella term for scientific and engineering aspects of computing. It starts with stating having extended its task to both Computer Science and Computer Engineering because it had concluded that no fundamental differences exist between the two fields in the core material (Denning, et al., 1989). The report distinguishes three major paradigms, or cultural styles, for the computing discipline: theory (rooted in mathematics), abstraction (rooted in the experimental scientific method), and design (rooted in engineering). The three cultural styles are fundamental for the discipline, which is described as a blend of interaction among theory, abstraction and design.

“Computing as a discipline” addresses the question, in which area of computing majors should be competent. The answer is cautious: discipline-oriented thinking should be the primary goal of every curriculum for computing majors, based on solid mathematical foundations. Teaching paradigms emphasizing inquiry and orientation in the computing literature are recommended, instead of lectures presenting answers.

“Computing as a discipline” pleads against a strong identification of computing with programming. It also states that every computing major should be competent in programming. The Task Force remarks, that the distinctions between the computing disciplines are embodied in the differences between programming languages, and recommends programming languages to be treated as a vehicle to access these distinctions.

Since 1989, the ACM/IEEE has jointly formed Task Forces that have published and iterated overview reports with recommendations for computing curricula. They appeared in 1991 and 2005. These reports have a triple ambition. They aim at (1) training the professionals requested by industry, (2) training the students’ intellectual skills necessary to find employment, enter the Master level and keep up with future developments, and (3) granting the development of the discipline of computing.

Historical Review And Epistemological Considerations

Over the years, the amount of sub disciplines has narrowed to 5 (ACM / IEEE, 2005): Computer Science, Computer Engineering, Information Systems, Information Technology, and Software Engineering. Undergraduate curriculum recommendations for each of these areas were published in 2005. Specific committees of ACM and IEEE representatives worldwide have updated them since then. CS2013 applies to Computer Science, one of the sub-disciplines of computing. It falls outside the scope of this investigation.

3.2.5 HYBRID COMPUTING CURRICULA

Until the 1970s, computers were used in professional setting in the industry, in academia and in big governmental institutions. The first computer users were programmers, but the user base changed as the costs of hardware lowered (Tedre, 2007). New interdisciplinary fields of study emerged, as Management Information Systems, combining computing topics with topics from other disciplines.

Today, all computing curricula are “hybrid” to a certain extent. Non-computing topics as Risk Management or Interpersonal Communication are taught in most undergraduate computing programs. But there also is a “core of computing”, common to all these programs. The last ACM/IEEE Overview Report, CC2005 (ACM / IEEE, 2005), provides 10 knowledge areas and 40 topics every Computing Curriculum should cover. The weight of these topics in single curricula can differ, and some topics might not be covered at all, but the clear intention of the Report is to describe a core, common to the five sub disciplines of computing.

Today, there also are undergraduate curricula covering a significant part of the core of computing, but omitting another significant part. Some examples are: Medical Information Systems (Dialogic & Matchcare, 2016), Business Analytics (Dialogic & Matchcare, 2016), Liberal Arts and Computer Science (Liberal Arts and Computer Science Consortium, 2007). We will adopt the Dutch researchers’ definition and will indicate as “hybrid” these curricula dedicating less than half of their program to topics belonging to the core of computing.

3.3 CULTURAL STYLES IN COMPUTING

The Task Force’s view on three cultural styles of computing (Denning, et al., 1989) was not new. In 1969, Peter Wegner had held a lecture at the annual meeting of the American Association for the Advancement of Science. Its title was: Three computer traditions (Wegner, Three computer traditions:

Computing Education in a Hybrid World

Computer technology, computer mathematics and computer science, 1970). Wegner objected to a single view of computing. CS, argued Wegener, is in part a scientific discipline concerned with the empirical study of a class of phenomena, in part a mathematical discipline concerned with the formal properties of certain classes of abstract structures, and in part a technological discipline concerned with the cost-effective design and construction of commercially and socially valuable products.

In a recent article (Tedre & Apiola, 2013) Tedre and Apiola discuss the different cultural styles of computing, or (in their terminology) three traditions of computing. According to the authors, these three traditions fulfill different roles in the development of the discipline of computing. The scientific tradition copes with the fundamental problem: does an abstract model fit the world? The engineering tradition copes with the question, how to translate abstract models into working artifacts. The theoretical tradition aims at building a coherent abstract framework supporting understanding of notions as algorithms, complexity, data structures. All authors agree upon the importance of intertwining the traditions in the computing practice.

Despite the statement in the 1989 report, that “no fundamental difference exists between the [...] fields in the core material”, founding the discipline on three traditions is a challenge. Tedre and Apiola (Tedre & Apiola, 2013) analyze the differences between the three cultural styles, and the implications of these differences implications for basic (K-12) computing education. They state that, however intertwined and overlapping, the three cultural styles are fundamentally different in their aims, in the status of the knowledge they pursue and in their methodological views.

The aim of the theoretical tradition is to produce coherent structures, in order to describe algorithms and cope with complexity. The scientific tradition aims at understanding the world by modeling phenomena (such as the weather, the mind) and testing the accuracy of the models. The engineering tradition aims at changing the world, by producing artifacts that fulfill social needs or desires.

In the theoretical tradition, knowledge is considered independent of what people may think. It is considered universal and is validated in its theoretical context. Also the scientific tradition claims to aim at value-free knowledge, which should be descriptive. This is opposed to the engineering tradition, in which knowledge is partially value-free (where it concerns physical constraints) and partially value-laden (where it concerns social needs and desires). The Body of Knowledge in the engineering tradition is partially descriptive and partially normative (know-how).

Historical Review And Epistemological Considerations

The theoretical tradition analyzes ideas. Propositions should be proven. The scientific tradition observes the world. Claims should be sustained by empirical results. The engineering tradition acts. It evaluates tangible products. Engineers must be able to act, even without having sufficient information to fully sustain the design of the artifact.

Tedre and Apiola's paper concerns K-12 computing education. The authors plead for aligning learning objectives with the corresponding computing traditions (expressed in pedagogic approaches and educational resources), because a mix would not result in effective educational interventions. If the learning objectives of the school concern construction processes, assessment tasks should not be propositional (quizzes), but procedural (assessment of design artifacts), because they better fit in the related tradition, which in this case is engineering. The authors stress that educators should understand all the traditions of computing. They warn for bias induced by hidden ethos elevating one of the traditions above the others, both inside schools as in university departments educating teachers.

Tedre and Apiola reflect on the role of computing traditions in educational design. The authors recommend matching the tradition with the learning objectives of educational units. In the end, it is an argument about the efficacy of educational interventions. It does not concern the learning objectives themselves, nor the goals and objectives of computing curricula.

3.4 DISCUSSION: CURRICULAR TRADE-OFFS

In this section we will reflect on hybrid computing education using the three pedagogical themes described earlier. Alignment of the three themes in combination with the limited space for computing in a hybrid curriculum gives rise to challenges. Below we will discuss three of these challenges: representing all cultures (cultural styles), fostering of discipline oriented thinking (goals and objectives) and the role of mathematics (goals and objectives, again). In our opinion, the trade-offs involved with curricular choices ought to be made explicit by authors of hybrid curricula. In Section 3.5 we will discuss two cases in more detail.

3.4.1 INCORPORATION OF THE THREE CULTURAL STYLES

In 1989, the Task Force on the Core of Computer Science described a common core of the discipline of computing. Content from each cultural style should be treated by all undergraduate computing curriculums. To prepare students for the future –which was uncertain— the 1989

Computing Education in a Hybrid World

recommendations included inquiry-based learning activities and orientation in the computing literature, instead of lectures presenting answers.

According to Tedre and Apiola, the three cultural styles not only entail different approaches to the discipline, but also respect different epistemological values. Should value-laden knowledge be accepted (as it is in the engineering culture) or should knowledge always be value-free? Should ideas be analyzed (theoretical culture) or evaluated empirically (scientific culture)?

An inquiry-based curriculum appears not to be neutral from this perspective. There is a risk that Institutions (schools, academies, departments) prefer one culture, one approach to research above others. Even an inquiry-based educational approach can canalize students' understanding of the discipline, towards values, consistent with the method of inquiry.

A possible direction for a solution is given by "Computing as a discipline" in stating "most of the distinctions in computing are embodied in programming notations". It suggests using differences between programming languages as a vehicle to discuss differences between approaches to computing. We will return to this in Section 3.5.2.

3.4.2 FOSTERING DISCIPLINE ORIENTED THINKING

The ACM/IEEE curricula are written from a triple ambition: to serve the students, serve the industry and grant the development of a (unified) discipline. In the light of Tedre and Apiola's findings, we are not optimistic about the possibility of fulfilling all the ACM/IEEE curricular ambitions by any of the undergraduate computing curricula. Especially the aims concerning overview of the discipline and employability appear to be scarcely compatible with each other. Mixing content from cultural styles will not automatically grant overview of the discipline, since knowledge is also rooted in the understanding of research methods. Nor will an investigative approach to education necessarily support discipline oriented thinking.

Employability and capability to keep up with future developments often suppose up-to-date knowledge in one of the sub disciplines and related research skills. Once acquainted with one of the cultural styles, the graduate is likely to pursue a career path compatible with it, reinforcing the chosen orientation.

If granting overview of the discipline is challenging for hybrid curricula, explicating boundaries becomes necessary.

Historical Review And Epistemological Considerations

3.4.3 THE ROLE OF MATHEMATICS

Both the 1989 Task Force and the authors of “Computing as a discipline” recognized the importance of a solid mathematical foundation to foster discipline oriented thinking.

We partially agree on the overall importance of mathematical thinking for computing professionals. Yes, the theoretical cultural style does support discipline oriented thinking, because it provides a common language to discuss computing constructs. It certainly does support formal reasoning. But Mathematics too is a vast discipline. Different branches of computing rely different branches of Mathematics. A course on continuous mathematics supports skills that are relevant to computer engineers, not necessarily to information scientists. The existence of one form of mathematical training, apt to foster overall understanding of computing, is debatable.

3.5 HYBRID CURRICULA: TWO CASES

In this section we will apply our findings to two existing hybrid programs: Liberal Arts and Computer Science and Front End Development, respectively. For each of these we will examine our pedagogical themes and the appearance of trade-offs.

3.5.1 LIBERAL ARTS AND COMPUTER SCIENCE

The Liberal Arts and Computer Science (LACS) curriculum (USA) (Liberal Arts and Computer Science Consortium, 2007), has its origin in a reaction to ACM’s Curriculum ’78, a strongly engineering oriented curriculum (Bruce, Cupper, & Drysdale, 2010) (Tedre, 2007). Liberal Arts undergraduate programs traditionally emphasize intellectual growth of students, rather than preparing them for a specific career. In 1984, a consortium of small Liberal Arts colleges offering computing degrees, was formed to discuss the problems these colleges had with the implementation of Curriculum ’78. The meetings resulted in the first Model Curriculum for a Liberal Arts Degree in Computer Science (LACS), which appeared in 1986.

LACS offers a Bachelor of Arts degree. Roughly 40% of the curriculum is dedicated to Computer Science, 5-10% to other science courses and the remaining 50-55% to humanities and social sciences (Liberal Arts and Computer Science Consortium, 2007). LACS is a hybrid computing curriculum.

It is interesting to see, what the LACS curriculum guidelines states about the role of mathematics. Mathematical education is considered part of the

Computing Education in a Hybrid World

Computer Science curriculum, for several reasons. Computing relies on mathematical objects (as sets, relations) and mathematical reasoning (logic, algorithms, correctness proofs); mathematical tools as probability and statistics allow analysis of software (Liberal Arts and Computer Science Consortium, 2007).

Cultural styles

LACS chooses a generalist approach and de-emphasizes specific technical details. It acknowledges the importance of mathematics in this – hybrid – curriculum. In terms of Tedre and Apiola’s work, the theoretical and the scientific view are combined in the LACS curriculum, but the engineering viewpoint is somehow overshadowed.

Goals and objectives

LACS strives to train generalists. The curriculum de-emphasizes specific technical details; its ambition is to develop student’s intellectual skills. The objective of the curriculum is to support durable intellectual growth, combined with (generalist, discipline oriented) helicopter view.

Trade-offs

The LACS curriculum explicitly makes concessions to its ambitions towards software development.

3.5.2 FRONT END DEVELOPMENT

We will refer to the discipline that is specialized in the development of multimodal user interfaces as Front End Development (FED), and to the corresponding professionals as Front End Developers (FEDs).

“A front-end developer specializes in building the front end, or client-side, of a web application, which encompasses everything that a client, or user, sees and interacts with. Front-end development is all about what’s visible to the user.” (Skilledup, 2016).

There is no consensus yet about an undergraduate curriculum for FED. In Amsterdam, FED is a major of the undergraduate curriculum Communication and Multimedia Design, in the domain Creative Technologies. FEDs are professional users of client-side technology. They work in interdisciplinary teams in developing projects. FEDs produce software that interacts with APIs, services and other ready-to-use software components, written by other computing professionals.

Cultural styles

FED is teamwork and aims to produce maintainable software products. For these reasons, we tend to include FED among the disciplines emphasizing primary ICT skills as described in section 3.1. Still, not all of

Historical Review And Epistemological Considerations

computing is relevant for FEDs. Most of the Front End Developer's work is visible to the user and can be tested directly. There is no need for correctness proofs in FED. FEDs need Mathematics to interpret quantitative research data. Acquaintance with basic abstract structures as sets and enumerations is needed, since every Front End communicates with other computer systems through abstract interfaces. In how far modeling skills are necessary for FEDs is subject to discussion. In terms of Tedre and Apiola's work, the engineering and the scientific aspects of computing are emphasized by this program, at the expenses of theoretical aspects.

Goals and objectives

FED is characterized by focus on craftsmanship, in the context of interdisciplinary developing teams and rapid technologic evolution. It strives at training competent manpower for the discipline of FED, who will be able to to comply with future developments in FED.

Instructional strategies

Front-end developing languages are introduced stressing their aims and their boundaries, conform the Task Force's remark on programming languages: "Software, written with HTML, performs behavior that is visible. It is not necessary to express intended outcomes in formal statements; the software can be tested. To support interaction with abstract agents, professional HTML code should respect conventional semantics. These are defined by a political process". This supports future FEDs' understanding of the scope of their craftsmanship, helps them to follow the evolution of their discipline, and to participate in that process.

Trade-offs

The choice for an engineering-based FED curriculum, at the expense of the theoretical content, has consequences. Graduates are not all-round software developers. Their sphere of activity is limited to the user interface. FEDs are trained in making Front-End artifacts, not in developing Front-End technology. They use and research the possibilities of existing technology. The question, whether FEDs should develop simple APIs is still open.

FED graduates are granted access to the Master's level in hybrid computing disciplines as Digital Communication and Media, not to Master's programs in Software Engineering.

3.6 CONCLUSIONS

We investigated the possibility to type hybrid computing curricula. Our conclusion is: refer to the complex epistemological background of computing to type hybrid curricula. It is inevitable for designers of hybrid

Computing Education in a Hybrid World

curricula to make choices. This approach gives insight in the related trade-offs; it is relevant for both educational institutions and accreditation boards.

We do think that some of the hybrid computing curricula can train developers, although bound to specific contexts, like FEDs. Though, it is fundamental to describe and stress the boundaries of such programs, and of the corresponding professions. Defining and tuning undergraduate computing curricula is not only a matter of training manpower requested by industry. We saw that the field of computing was shaped by a joint effort of industry, academia and governmental institutions, and urge designers of hybrid curricula, to take responsibility for the definition of new disciplines. Trade-offs, involved by curricular choices, should explicitly be discussed in broad communities including: related computing disciplines in academia, the industry and the public authorities.

Following Tedre and Apiola's observations, we also recommend that all educators of computing topics understand the complex nature of computing. This applies in particular to those lecturing in hybrid curricula, a condition that is not always met. We address educators, to warn them against a sloppy approach to computing: (1) lecturing the outcomes without the method would not result in durable knowledge; (2) lecturing technology without mentioning its trade-offs would not allow students to position themselves as professionals in the dynamic field of computing; (3) lecturing the method without the philosophical motivations would mean not taking students seriously enough to invite them to join the conversation about their own professional future.

4 Computing Curricula in Dutch Universities of Applied Sciences

4.1 INTRODUCTION

In this section, we investigate similarities and differences between the Dutch framework for computing education at the Universities of Applied Sciences (the HBO-ICT framework) and international frameworks that were designed to describe the discipline of computing. Our goal is twofold: (1) to understand Dutch tertiary professional education, and (2) to understand aims and purpose of its undergraduate computing curricula and their relation to hybrid curricula.

We are not interested in the individual curricula, but in what they have in common. Dutch institutions for professional educational refer to national and international frameworks to build and validate their curricula. We will examine the following frameworks.

- (1) ACM/IEEE recommendations for the undergraduate computing curricula,
- (2) the European E-Competence Framework e-Cf 3.0 ,
- (3,4) the last two versions of the HBO-ICT framework for computing education at the Dutch Universities of Applied Sciences, the "Domeinbeschrijving Bachelor-ICT",
- (5) The framework for the new hybrid domain in Dutch Universities of Applied Sciences, HBO-Creative Technologies

Frameworks 3 and 4 form the basis for Dutch undergraduate computing curricula at Universities of Applied Sciences. In some cases, hybrid curricula are also covered by these frameworks' descriptions. In such cases, we will briefly discuss the position the framework gives to hybrid computing curricula.

In 2014, a new framework HBO-Creative Technologies was published in the Netherlands. Dutch programs Creative Technologies refer to this framework (5) at the present.

4.2 FRAMEWORKS FOR CURRICULUM RECOMMENDATIONS

In the Netherlands, concerning professional undergraduate education, basically the following frameworks for curriculum recommendations exist:

Computing Education in a Hybrid World

(1) ACM/IEEE curriculum recommendations series (Denning, et al., 1989) and its most recent overview report CC2005 (ACM / IEEE, 2005)

(2) The European e-Competence Framework e-CF 3.0 (European Committee for Standardization, 2014);

(3,4) The HBO-ICT framework for computing education at the Universities of Applied Sciences. The last edition of the HBO-ICT framework dates from 2014; our research in Part II concerns students, enrolled in curricula that were designed prior to 2014. For that reason, we will investigate the last two editions of the HBO-ICT framework: 2009 (Schagen, Kwaal, Leenstra, Smit, & Vonken, 2009) and 2014 (Valkenburg, et al., 2014).

(5) We will involve the curriculum framework HBO Creative Technologies (Domein Creative Technologies, 2014) because it describes hybrid programs. Some of them were addressed by the 2009 version of the HBO-ICT framework, but fall outside the scope of HBO-ICT today. We remark that this first version of the HBO framework Creative Technologies is not a fully mature curriculum framework yet. It is more a description of curricular aims.

4.2.1 ACM/IEEE SERIES

The origin of the ACM/IEEE curriculum recommendations is described extensively in section 3.2.4. These recommendation frameworks are written and updated by worldwide committees ("Task Forces"), mainly of scholars. Industry and (North American) governmental organizations are consulted in the process. The members of these Task Forces are recruited among the members of the professional organizations for computing professionals, ACM and IEEE. The first joint ACM/IEEE curricular Task Force was established in 1989 in North America. Today, the coverage is global and so is the range of the ACM / IEEE Curriculum Recommendations. In particular, they are determinant for Dutch Universities (QANU, 2014).

Today, the ACM/IEEE curriculum recommendations series is articulated as follows: an overview report describing the discipline of computing and, for each sub-discipline, reports with curriculum recommendations. We will mainly refer to the first Task Force's report (Denning, et al., 1989) and the most recent overview report Computing Curricula 2005 (CC2005) (ACM / IEEE, 2005), describing ten elements that "Any reputable computing degree programs should include". "Computer programming" is one of them. Figure 4-1 shows its description.

Computing Curricula in Dutch Universities of Applied Sciences

2) A foundation in the concepts and skills of computer programming. The foundation has five layers:

- a) an intellectual understanding of, and an appreciation for, the central role of algorithms and data structures;
- b) an understanding of computer hardware from a software perspective, for example, use of the processor, memory, disk drives, display, etc.
- c) fundamental programming skills to permit the implementation of algorithms and data structures in software;
- d) skills that are required to design and implement larger structural units that utilize algorithms and data structures and the interfaces through which these units communicate;
- e) software engineering principles and technologies to ensure that software implementations are robust, reliable, and appropriate for their intended audience.

Figure 4-1 CC2005, description of Computer Programming

To illustrate our findings, we will refer to the most recent updates of the curriculum recommendations reports, which are relevant for the Dutch professional undergraduate curricula. These are: CS2013 concerning Computer Science (ACM / IEEE, 2013), SE2014 concerning Software Engineering (ACM/IEEE, 2014), and IT2017 concerning Information Technology (ACM / IEEE, 2017).

A typical report for curriculum recommendations will describe the correspondent sub discipline of computing and its evolution since the last report with curriculum recommendations. It will state its view on education. It will list essential topics (that all undergraduate curricula in that specific sub discipline should address), and electives or supplementals (topics that can be included in the curriculum). Finally, it will provide examples of curricula.

Hybrid computing curricula traditionally fell outside the scope of ACM/IEEE curriculum recommendations. IT2017 (ACM / IEEE, 2017) includes, for the first time, models for the implementation of IT programs in different contexts. One of these, the “IT is a concentration in a larger degree program”, concerns (in our terminology) hybrid curricula. The guideline is: cover the essential part of the IT topics, which requires approximately one year’s program.

Computing Education in a Hybrid World

4.2.2 E-CF

The European e-Competence Framework (e-CF) is an industry standard. It provides for 40 competences “as required and applied in the ICT workplace” (European Committee for Standardisation, 2014). Its development was supported by the European Commission. The Framework aims to enhance understanding of ICT-professionals’ competences and roles across Europe. It is a component of the European union’s strategy for e-Skills in the 21st Century.

e-CF is the result of an iterative process. The initiative was taken by representatives of the European Industry such as Airbus, Michelin and BITKOM. One educational Institution participated (Politecnico di Milano) and the United Kingdom’s sector skills council e-Skills UK, which is a network of employers (UK, sd). The initiative was supported by the European Community (European Committee for Standardisation, 2014). In 2006, existing computing job profiles frameworks across Europe were compared and found fundamentally different (Valkenburg, et al., 2014). Two years multi-stakeholder (ICT and human resources experts’) work followed, from multiple organisation levels. (European Committee for Standardisation (2), sd). The outcome was version 1.0 of e-CF, published in 2008. The framework was extended and updated in 2010 and in 2013, taking into account the stakeholders’ feedback and application experiences. It became a European standard in 2016. European governmental organisations also participate in e-CF today, including the Dutch Ministry of Economic Affairs.

e-CF defines competence as “a demonstrated ability to apply knowledge, skills and attitudes for achieving observable results” (European Committee for Standardization, 2014, p. 5). Strictly speaking, e-CF does not provide for curriculum recommendations. But e-CF 3.0 explicitly mentions in its foreword that the Framework also was created for application by education institutions, including higher education.

The framework is structured in four dimensions. Dimension 1 refers to stages in the ICT business process plan (Plan-Build-Run-Enable-Manage). Dimension 2 consists of a set of e-competences for each stage listed in dimension 1, 40 in total. Dimension 3 lists proficiency levels for each e-competence, and refers to the European Qualification Framework EQF (European Commission, 2005). EQF’s scope is larger than e-Cf. We will look at e-CF’s proficiency level 3, corresponding with the Bachelor’s level. Dimension 4 gives samples of knowledge and skills related to the e-competence; these samples are not intended to be exhaustive. You will find

Computing Curricula in Dutch Universities of Applied Sciences

the description of e-competence B1 (Build – Application Development) in Figure 4-2.

European e-Competence Framework		3.0				
Dimension 1 e-Comp. area	B. BUILD					
Dimension 2 e-Competence: Title + generic description	B.1. Application Development Interprets the application design to develop a suitable application in accordance with customer needs. Adapts existing solutions by e.g. porting an application to another operating system. Codes, debugs, tests and documents and communicates product development stages. Selects appropriate technical options for development such as reusing, improving or reconfiguration of existing components. Optimises efficiency, cost and quality. Validates results with user representatives, integrates and commissions the overall solution.					
Dimension 3 e-Competence proficiency levels e-1 to e-5, related to EQF levels 3 to 8	Level 1 Acts under guidance to develop, test and document applications.	Level 2 Systematically develops and validates applications.	Level 3 Acts creatively to develop applications and to select appropriate technical options. Accounts for others development activities. Optimizes application development, maintenance and performance by employing design patterns and by reusing proved solutions.	Level 4 -	Level 5 -	
Dimension 4 Knowledge examples <i>Knows/aware of/familiar with</i>	K1 appropriate software programs/modules K2 hardware components, tools and hardware architectures K3 functional & technical designing K4 state of the art technologies K5 programming languages K6 Power consumption models of software and/or hardware K7 DBMS K8 operating Systems and software platforms K9 Integrated development environment (IDE) K10 rapid application development (RAD) K11 IPR issues K12 modeling technology and languages K13 interface definition languages (IDU) K14 security					
Skills examples <i>is able to</i>	S1 explain and communicate the design/development to the customer S2 perform and evaluate test results against product specifications S3 apply appropriate software and/or hardware architectures S4 develop user interfaces, business software components and embedded software components S5 manage and guarantee high levels of cohesion and quality S6 use data models S7 perform and evaluate test in the customer or target environment S8 cooperate with development team and with application designers					

Figure 4-2 e-Competence Application Development (B1) of e-CF 3.0

In a separate publication (CEN), e-CF describes 23 iconic profiles of computing professionals, the e-competences required to fulfill these professional roles, and the corresponding levels of proficiency. These are linked to the European Qualification Framework EQF. We remark that

Computing Education in a Hybrid World

“digital media specialist”, a profile that is considered hybrid in the Netherlands, is one of these profiles.

4.2.3 DUTCH BACHELOR OF ICT FRAMEWORKS

The Netherlands has a dual system of tertiary (post-secondary) education conferring Bachelor’s and Master’s degrees (HBO-Raad, 2009). The academic track focuses on research and development of knowledge areas. The application of knowledge is delegated to highly educated professionals – nurse practitioners, physiotherapists, teachers, social workers, hydraulic engineers – who are not educated in pre-university schools and academia, but in a professional track, whose tertiary stage is called Hoger Beroeps Onderwijs (HBO). Institutions offering these programs are called Universities of Applied Sciences. HBO is organized in sectors; technology is one of them. The technology sector used to be organized in four domains: ICT, Engineering, Built Environment and Applied Science. Maritime Operations and Creative Technologies were added in 2015.

A typical undergraduate HBO curriculum spans over 4 years and always includes a professional internship (Commissie Accreditatie Hoger Onderwijs, 2011). The track prepares for entering the labor market; it is not primarily intended to access academic Master courses. For that reason, most Universities require the HBO graduate to accomplish a bridging program of 6-12 months before enrolling for a Master’s degree. Undergraduate HBO programs are defined by educational institutions, and are subject to accreditation by the Dutch Flemish Accreditation Board NVAO.

The Dutch managers of HBO programs in computing at Universities of Applied Sciences are organized in the HBO-I association. This association periodically publishes a description of the qualifications undergraduate computing curricula strive at, in collaboration with representatives of Dutch industry. Like the ACM/IEEE computing curricula, HBO-I distinguishes different sub disciplines of computing. These are: Informatica, Technische Informatica and B-ICT. This investigation concerns the global framework for computing curricula at the Dutch Universities of Applied Sciences or, in HBO-I’s terminology, the “Domeinbeschrijving HBO-ICT. We will use “HBO-I” to indicate the association, “HBO-ICT framework” to indicate the global curriculum framework and “HBO-ICT” to indicate the educational domain, i.e. all undergraduate computing programs offered by Dutch Universities of Applied Sciences.

The HBO-ICT framework is a systematic, exhaustive description of the final qualifications for Dutch HBO programs Bachelor of ICT. It is task-based,

Computing Curricula in Dutch Universities of Applied Sciences

and lists the typical professional tasks, performed by Bachelor-ICT graduates. However, no HBO program in computing will cover all the matching learning goals. One of the framework's aims is to support educational institutions in explicating which part of this description is covered by their curriculum.

Most Dutch educational institutions refer to the HBO-ICT framework to build and validate their curricular choices, and a number of these institutions explain how far these choices match e-CF. For example Rotterdam University of Applied Sciences and HU University of Applied Sciences Utrecht follow this strategy. Avans University of Applied Sciences acts differently. It compared the HBO-ICT framework, SE2014 and e-CF, and decided to build its computing curriculum on e-CF instead of the HBO-ICT framework or SE2014.

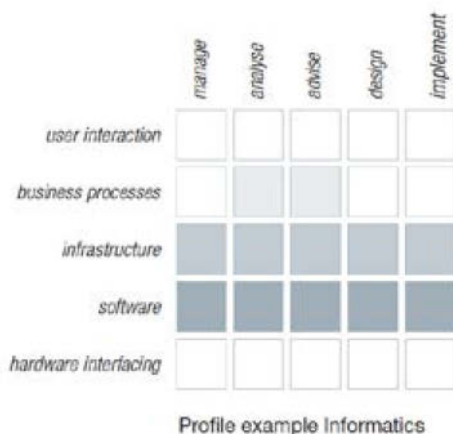


Figure 4-3 Description of a possible HBO-ICT program. Darker colors indicate higher levels of proficiency

The HBO-ICT framework is structured in 3 dimensions. Dimension 1 consists of 5 software lifecycle activities (Manage, Analyze, Advise, Design, and Implement). Dimension 2 consists of 5 architectural layers (User Interaction, Business Processes, Infrastructure, Software and Hardware Interfacing). Dimension 3 consists of 3 levels of proficiency, linked to different kinds of contexts: stable - predictable - unpredictable); these levels are linked to e-Cf levels.

HBO-ICT programs can show the emphasis of curricula by stating which levels of proficiency graduates will attain for all relevant combinations of

Computing Education in a Hybrid World

lifecycle activity and architectural layer. An example of such descriptions is included in Figure 4-3 (Valkenburg, et al., 2014), English version.

For the reasons stated in the introduction of this section we will look at the last two editions of the HBO-ICT framework: HBO-ICT 2014 (Valkenburg, et al., 2014) and HBO-ICT 2009 (Schagen, Kwaal, Leenstra, Smit, & Vonken, 2009). These editions of the framework have similar structures, but the 2009 edition of the HBO-ICT framework also includes examples of professional profiles, with the corresponding tasks. You will find the professional tasks associated with the dimension “Software” in Figure 4-4.

Unlike the ACM/IEEE recommendations, the HBO-ICT framework gives no indications of the efforts that should be devoted to specific topics, “lectures” or “curricular hours”.

4.2.3.1 Similarities and differences HBO-ICT 2009 & 2014

Both editions of the HBO-ICT framework describe the domain of ICT by listing tasks that starting computing professionals should be able to fulfill. Both descriptions are written by representatives of educational institutions, in collaboration with representatives of Dutch industry. They both refer to international frameworks. They both aim to mark the boundaries of the domain of ICT. There also are differences, though.

The 2009 edition provided a dynamic description of the domain of ICT. It strived at supporting educational institutions to keep up with developments in the domain, by allowing specialized curricula as Business Informatics or Communication and Multimedia Design to fit in the description. From 2014 onwards, the domain of ICT is not longer seen as fluid. Emphasis has shifted from specialization to standardization. In 2014, there seems to be more consensus on the “profession of ICT” and its boundaries in the Netherlands. The 2014 framework reflects this consensus. Today, Communication and Multimedia Design is not seen as a program of the domain of ICT anymore, but of Creative Technologies.

In the 2009 spirit of specialization, knowledge too was seen as fluid. Bodies of Knowledge (BoKs) were seen as context specific. Institutions offering computing programs that did not have developed a BoK yet, were urged to develop one. The framework addressed the question, how students should be prepared to keep up with future developments. Students should learn to read professional literature. In 2014, teaching students to read professional literature is not sufficient anymore. The 2014 framework sees graduates as contributors to the professionalization of the computing

practice. The framework puts more emphasis on (applied) research, and on a theoretical foundation common to all graduates.

SOFTWARE	Event	Manage	Analyse	Advise	Design	Implement
	3	Implement configuration, change and release management.	Carry out a requirement analysis for a software system with various stakeholders, in the context of existing systems. Identify integration and migration issues. Define acceptance criteria on the basis of quality characteristics and a performed risk analysis.	Advise on the choice of software architecture or software frameworks, in which cost and quality aspects such as scalability, performance, security and availability play a part. Advise on the format of a software development process, including the test process.	Set up a software architecture for a software system, consisting of both existing and new systems, taking into account both quality aspects and stakeholders. Draw up a test strategy for system tests.	Build and make available a software system in line with existing systems and on the basis of the designed architecture, using existing frameworks. Using test automation when performing tests.
	2	Layout, manage and use a software factory to support software development within teams. Apply principles to manage and monitor a software development process.	Carry out a requirement analysis for a software system with various stakeholders, taking into account the quality characteristics. Perform an analysis to formulate and validate functionality, design, interfaces and the fit, of an existing system or component. Set up an acceptance test on the basis of quality characteristics.	Advise on possible purchases and subsequent selection of existing software or components when developing software systems in which the cost aspect features. Advise on an architecture component of a limited software system. Advise on the use of prototypes when validating requirements.	Set up a functional design for a software system, taking into account the existing components and libraries, and using quality criteria of design. Determine the quality of the design, for example by testing or prototyping, taking into account the formulated quality characteristics. Draw up test designs according to a predetermined test strategy.	Build and make available a software system consisting of multiple subsystems, and using existing components. Integrate software components within an existing system, whilst monitoring the integrity and system performance. Perform regression tests. Perform and report on unit, integration and system tests.
	1	Layout and use a management system to support software development within a team setting.	Collect and validate functional requirements for a software system with one stakeholder and according to a standard method. Define acceptance criteria for the above-mentioned functional requirements.	Provide recommendations regarding specific requirements of a software system, on the basis of research into existing, similar systems.	Design a software system with modeling techniques using a standard method.	Build and test an elementary software system and make it available.
	Manage	Analyse	Advise	Design	Implement	

Figure 4-4 Dimension "Software" of HBO-ICT 2014

4.2.4 HBO CREATIVE TECHNOLOGIES

The first description of undergraduate programs Creative Technologies was published in 2014, by managers of pre-existing hybrid HBO-programs with a creative component. The domain was created to better serve the economic sector Creative Industries, which is one of the economic top sectors in the Netherlands. Creative Industries differs from other technology-driven domains as ICT because it mostly consists of small companies, where interdisciplinary teams apply cutting-edge digital technologies in small-scale projects (Domein Creative Technologies, 2014). Typical professional products are Serious Games and Smart Fashion, where technology, psychology and art fuse together. One of the graduates' core competences is: acting in multi- and interdisciplinary teams.

Creative Technologies covers three programs: Communication and Multimedia Design, Creative Media & Game Technologies and Fashion & Textile Technologies. The curriculum framework is a competence-based description of what the three programs have in common. In our analysis, we

Computing Education in a Hybrid World

will look at the competences related to computing. The framework does not use the word “computing”, though. It uses the term “(digital) technology”. By doing so, it seems to open the door for other new technologies than computing in the future. However, no other technologies than digital technologies are mentioned at the present.

In section 4.3.2.1, we saw that in the Netherlands, the position of HBO-ICT towards hybrid programs has changed. Communication and Multimedia Design (CMD), the program training Digital Media Specialist, was considered as a possible new variant of a computing curriculum in the HBO-ICT 2009 framework but not anymore in the 2014 framework. e-CF adopts a different position. One of the 23 standardized European ICT profiles (CEN) is “Digital Media Specialist”. For this professional figure, undergraduate level of proficiency is requested for the following e-CF competences: B.1 (Application development) and B.4 (Solution deployment). This is compatible with the guidelines of Creative Technologies. e-CF seems less strict than HBO-ICT in its definition of ICT.

4.3 MODELS FOR COMPARISON

The aim of this chapter is to understand Dutch computing education at the Universities of Applied Sciences. In the next section, we will compare the HBO-ICT framework for computing curricula with the ACM/IEEE computing curricula and with the European e-Competence Framework.

Comparing curricular frameworks is a complex task. It entails the choice for a terminology covering differences and similarities between curricula at a high level of abstraction. But national educational systems differ; policymakers express themselves in national languages, refer to different systems and use different terminology.

In 2011, an ITiCSE working group elaborated a coding system for comparing Computer Science Education research in secondary education: the Darmstadt model (Hubwieser, et al., 2011). The Darmstadt model was written to bridge differences between national systems. Its origin is well documented. The model was elaborated to set the standard for a terminology to discuss educational systems. It is sound and very complete. But it also was designed for secondary education. Secondary and tertiary education are regulated differently. Curricular aims, goals and objectives of secondary education (as defined by UNESCO, (UNESCO, 2018)) are usually set by governmental institutions. Tertiary education sets its own goals and objectives. For that reason, we will also investigate an alternative approach.

Computing Curricula in Dutch Universities of Applied Sciences

We will compare the Darmstadt model with van den Akker's work on (generic) curriculum design research (van den Akker, Building bridges - how research may improve curriculum policies and classroom practices, 2010). As a result, we will expand the Darmstadt model with one aspect, central in van der Akker's model for curriculum description: the rationale.

4.3.1 THE DARMSTADT MODEL

The Darmstad model is an elaboration of P. Heimann's Berlin Model (Hubwieser P. , 2013), which was designed to support teachers in evaluating their educational efforts. In 2011, the ITiCSE working ggroup chaired by Hubwieser took the Berlin Model as a starting point to compare research in (secondary) computing education. But educational research does not only concern classroom activities. It can also focus at individual students' accomplishments, or at national educational programs.

Through a process of coding papers, the workgroup further specified some terms to the Berlin Model and added some others. Finally, the workgroup agreed on a new model, where the top dimensions of the Berlin Model were maintained as an ordinal scale (preconditions of learning: 1 / actor's decision area: 2 / consequences of measures: 3), and two dimensions were added: an ordinal scale specifying the reporting persons' range of influence (student: 1, classroom: 2, school: 3, region: 4, state: 5, country: 6, international: 7) and a nominal scale with relevant areas in educational research (outcomes/effects, media, extracurricular activities, teaching methods, examination/certification, curriculum issues, knowledge, intentions, motivation, teacher qualification, policies, socio-cultural factors, educational systems). The result of this process is the 3-dimensional Darmstadt model (Figure 4-5).

Computing Education in a Hybrid World



Figure 4-5 The Darmstadt model, v 1.1

We refer to the 2013 description of the Darmstadt Model (Hubwieser P. , 2013). The educational areas that are most relevant for our purpose are “knowledge” and “intentions”. “Intentions” includes competencies (defined by the ITiCSE workgroup as *the needs of the customers or the desired outcomes*), learning objectives (defined by the workgroup as *the aims of the teaching persons*) and educational standards (Hubwieser, et al., 2011).

Hubwieser (Hubwieser, et al., 2011) remarks that “standardization of the subject of informatics runs far behind traditional subjects like mathematics”. We agree. All the frameworks we investigate are meant to standardize the outcomes of undergraduate computing education. But, as Hubwieser remarks, the definition of educational concepts is poorly developed. We will list among “educational standards” the global criteria for qualification the Bachelor’s level, specific for computing, identified in the frameworks.

We will compare the curriculum frameworks along these “relevant areas” of the Darmstadt Model: “knowledge”, “competencies”, “learning objectives” and “educational standards”.

4.3.1.1 Discussion

Despite the dimensions added by the ITiCSE workgroup, we cannot adopt the Darmstadt model as it is for our purpose. We still miss terms. When the ACM/IEEE Task Force on the Core of Computing argues that

Computing Curricula in Dutch Universities of Applied Sciences

computing is “a” (unified) discipline that all computing professionals should know, when European employers choose competences as the basis for the definition of professional roles, they do more than solving a practical problem. They choose a position in how to address the problem. Such choices are not covered by the Darmstadt model.

One could argue that the approach to computing is a matter of “general opinion towards ICT”, one of the Darmstadt Model’s Socio-Cultural factors. But analysis of the text (Hubwieser, et al., 2011) reveals that these “Socio Cultural Factors” should be considered fixed qualities as age and gender. We also considered the possibility to code this aspect between the “Policies”. We rejected this option because in the Darmstadt model, the term “Policies” refers to modes of action. We are interested in the reasons behind these modes of action.

We will borrow a consistent part of the terminology for our comparison from the Darmstadt model, but we still miss one relevant area for the comparison of curriculum recommendations: the approach that resulted in curriculum recommendations.

4.3.2 VAN DEN AKKER’S CURRICULAR SPIDER WEB

According to van den Akker (van den Akker, Building bridges - how research may improve curriculum policies and classroom practices, 2010), curriculum design is a matter of balance between the curriculum’s components. These components are: Rationale, Aims and Objectives, Content, Learning Activities, Teacher Roles, Materials and Resources, Grouping, Location, Time and Assessment. Van den Akker visualizes these components as a spider web, to emphasize the curriculum’s complexity and its vulnerability.

Van den Akker puts the Rationale at the center of this spider web (Figure 4-6). The rationale is the curriculum designers’ position statement about the curriculum’s motivation. It expresses (1) The academic and cultural backgrounds of the program (2) Problems and issues, relevant for society (3) Elements of vital importance or interest for the learners themselves. Curriculum design, argues van den Akker, is a matter of choices. To keep the curriculum balanced and consistent, inevitable choices in curriculum design should refer to the Rationale.

In his earlier work (van den Akker, Curriculum perspectives, an introduction, 2004), van den Akker distinguished four levels of curricular activities as policymaking, design, development, implementation and evaluation. These levels were: *macro* (system/society/nation), *meso*

Computing Education in a Hybrid World

(school), *micro* (classroom) and *nano* level (individual). Three curriculum's components are most important at the *macro* level, argued the author. These are the Rationale, the Aims and Objectives and the Content. Later, van den Akker added a higher level (van den Akker, Building bridges - how research may improve curriculum policies and classroom practices, 2010): the *supra* level (international / comparative). For our comparison, we will look at the "Knowledge", the "Aims and Objectives", and at the "Rationale".

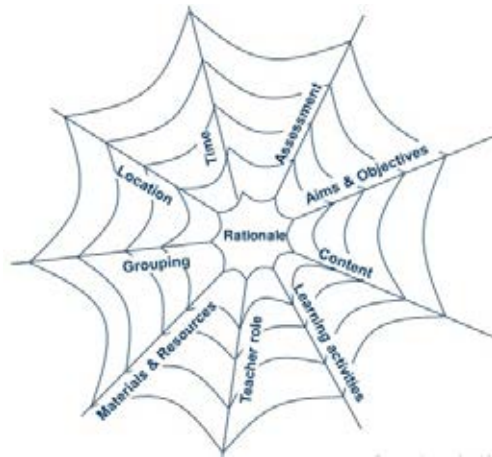


Figure 4-6 van den Akker's curricular spider web

4.3.3 5 ASPECTS

The act of publishing curriculum recommendations takes place in what the Darmstadt Model labels: "the author's Decision area". The 4 frameworks we discuss (ACM/IEEE curriculum recommendations, European e-Competence Framework, HBO-ICT frameworks) can all be located at level 2 of the "Berlin Model Top Dimensions" scale, the "Decision Areas". All frameworks have national or international scope, which corresponds with levels 6 or 7 of the Darmstadt Model's "Range of Influence".

We will look at some of the "Educational relevant Area's". Van den Akker's components "Aims and Objectives" and "Content" can be traced here, resp. as "Intentions" and "Knowledge". Van den Akker's "Aims & Objectives" answers the question "towards which goals are they learning?" It matches the Darmstadt Model's "Intentions", which includes competencies (defined by the ITiCSE workgroup as *the needs of the customers or the desired outcomes*), learning objectives (defined by the

Computing Curricula in Dutch Universities of Applied Sciences

workgroup as *the aims of the teaching persons*) and educational standards (defined as: qualification criteria, specific for computing in section 4.3.1). Van den Akker's "Content" answers the question "What are they learning?" In the Darmstadt Model, it corresponds with the "Knowledge" area. We will add van den Akker's Rationale answering the question "Why are they learning?"

Summarizing, we will compare these frameworks' following aspects:

- Rationale (answer to "Why are they learning?")
- Intentions: Learning Objectives (aims of the teaching persons)
- Intentions: Competencies (needs of the customers, desired outcomes)
- Intentions: Educational standards (criteria for qualification, specific for computing)
- Knowledge (answer to "What are they learning?")

4.4 CURRICULAR FRAMEWORKS COMPARED

4.4.1 RATIONALE

The curriculum's Rationale answers the question "Why are they learning?"

4.4.1.1 ACM/IEEE series

The rationale of the ACM/IEEE curriculum recommendation series is stated in the report that announced the ACM/IEEE joint Task Forces on Computing Curricula (Denning, et al., 1989). The goal of education, states the report, is to gain competence in a domain. The domain of computing is a complex domain, because computing is an intellectual discipline and a rapidly evolving discipline. The report divided it in sub-disciplines, supported by research communities. According to the report, the goal of undergraduate computing education is to foster discipline oriented thinking. Graduates in all the computing sub-disciplines should be acquainted with the common core of computing, which is defined by a joint effort of academia and industry. Undergraduate programs should foster the sub disciplines' understanding of each other's view on computing, and ensure further development of a united discipline of computing.

Computing Education in a Hybrid World

4.4.1.2 e-CF

The European e-Competence Framework was born as an effort to improve transparency of ICT practitioners' professional roles and degrees across Europe. e-CF is an industrial standard, not a framework for curriculum design. But it does refer to the European Qualification Framework EQF, and the reference "has been systematically developed to enable consistent interpretation of the EQF in the ICT workplace environment" (European Committee for Standardisation (2), sd). e-CF is meant as a reference for European computing education in general and tertiary education in particular. The e-CF's rationale is: to prevent shortage of qualified ICT manpower, by establishing a common language to express professional roles and competencies across Europe. This will support mobility of qualified manpower and define competencies that can serve as a guideline for the development of educational programs.

4.4.1.3 HBO-ICT 2009

The HBO-ICT framework is a task-based, exhaustive description of the final qualifications for Dutch HBO programs Bachelor of ICT. The 2009 framework strived at empowering Dutch institutions, offering tertiary professional education in computing, to design new programs. It provided these institutions with an instrument, suited to describe new or specialized computing curricula, in order to keep their education up-to-date. The 2009 edition aimed at supporting specialization.

The framework responded to the rapid sequence of upcoming technology by providing a dynamic description of the domain of ICT, not based on technology (that changes too rapidly) but on professional tasks. The description strived at compatibility with international standards. It referred to the first edition of e-CF and acknowledged existence of the ACM/IEEE curricular frameworks.

4.4.1.4 HBO-ICT 2014

The 2014 edition of the HBO-ICT framework emphasizes standardization. Its aim is to describe learning outcomes of Dutch professional Bachelor of ICT programs. The description matches e-CF 3.0. Conforming to this description enables educational institutions to train ICT professionals requested by Dutch and European industry. The framework also aims to train professionals who will contribute to the professionalization of the computing practice.

Computing Curricula in Dutch Universities of Applied Sciences

4.4.1.5 HBO-Creative Technologies

The HBO-sector Creative Technologies was created to establish educational programs, requested by the Dutch economic top sector Creative Industries. Its aim is to train professionals for companies that can act as motors for innovation.

4.4.2 INTENTIONS: LEARNING OBJECTIVES

Learning objectives are defined as “the aims of the teaching persons”. This section will discuss the educational aims of the frameworks.

4.4.2.1 ACM/IEEE series

The main learning objective of the ACM/IEEE curriculum recommendations is: support students’ intellectual development, in order to: (1) raise new generations of researchers in the field of computing and (2) equip students with enough background to keep up with developments of the discipline of computing.

4.4.2.2 e-CF

e-CF is not a curriculum framework. But it explicitly also “was created [...] for education institutions and training bodies including higher education” (European Committee for Standardization, 2014). The learning objectives can be summarized as: providing access to the pan-European labor market to graduates.

4.4.2.3 HBO-ICT 2009

The HBO-ICT 2009 framework’s educational aim was to train students’ skills and understanding, needed to fulfill standardized professional tasks. In order to keep up with future developments in the domain of computing, students should learn to read professional literature and develop aptitude for learning.

4.4.2.4 HBO-ICT 2014

The HBO-ICT 2014 framework is written to train students’ skills and understanding, needed to fulfill standardized professional tasks. The framework remarks that fulfilling professional tasks in the domain of computing always requires (a) the acquisition of knowledge and (b)

Computing Education in a Hybrid World

contributing to the professionalization of the computing practice. Therefore, students should develop skills for applied research.

4.4.2.5 HBO-Creative Technologies

The curriculum recommendations of HBO-Creative Technology aim at training professionals to work in interdisciplinary teams, where new digital technologies are adopted to design and develop groundbreaking, people-oriented applications.

4.4.3 INTENTIONS: COMPETENCIES

In this context, competencies are defined as “needs of the customers, desired outcomes”. The customers are: Master’s programs, industry and students. This section discusses the question: “which customer needs do the frameworks meet?”

4.4.3.1 ACM/IEEE series

The ACM/IEEE curriculum recommendation series aim at (1) providing Academia with students, qualified to enter the Master’s level and (eventually) doctoral programs; (2) providing industry with skilled workforce in a rapidly changing world.

Plessius and Ravesteyn (Plessius & Ravesteyn, 2016) compare e-CF with ACM’s description of the domain of IT (or, in our terminology: the domain of computing). They remark that ACM’s description stresses a more technical definition of the domain. In their opinion, it does not fully do justice in to the more business-oriented scope of e-CF.

IT2017, the most recent report in the curriculum recommendations Series, has a revolutionary approach. It “diverges from existing computing curricula guideline documents by focusing on competency instead of knowledge expectations” (ACM / IEEE, 2017, p. 15). IT2017 describes learning outcomes in terms of performances, i.e. “actions revealing understanding”. The IT2017 workgroup’s aim is to encourage academic departments to forge working collaborations with employers, targeted at offering students meaningful learning contexts.

4.4.3.2 e-CF

Competencies are the core of e-CF. The framework provides European professionals, educational institutions and industry up-to-date descriptions of competencies that are relevant for the computing profession and uniform

Computing Curricula in Dutch Universities of Applied Sciences

across the European countries. e-CF establishes a common language to express professional roles and competencies across Europe

4.4.3.3 HBO-ICT 2009

The 2009 edition HBO-ICT framework was written to clarify differences and similarities between Bachelor of ICT programs in the Netherlands. The aim was to support students in their educational choices and industry in grasping the curricular aims of educational institutions. To do so, it listed many examples of professional situations recent graduates could encounter, and linked these to the professional tasks, listed in the framework.

4.4.3.4 HBO-ICT 2014

The 2014 edition of the HBO-ICT framework strives at the definition of standard professional tasks and levels of proficiency, requested for fulfilling standardized professional roles. It refers to e-CF 3.0. The framework enables Dutch institutions, offering Bachelor of ICT programs, to explain the focus of their program.

4.4.3.5 HBO-Creative Technologies

This first edition of the curriculum recommendations focuses on the industry's requests: a broad-based education (necessary to act in interdisciplinary teams), and knowledge of cutting-edge (digital) technologies.

The authors expect to offer a valuable alternative to a group of students who otherwise would have to choose between a technology oriented program and a creative one.

4.4.4 INTENTIONS: EDUCATIONAL STANDARDS

We have defined educational standards as “global criteria for qualification at the Bachelor’s level, specific for computing”.

Strictly speaking, none of the frameworks we are investigating sets educational standards. But they all provide in authoritative descriptions of what society can expect from graduates in computing, descriptions that are referred to in accreditation processes.

This section discusses the question: “what can society expect from graduates in computing?”

Computing Education in a Hybrid World

4.4.4.1 ACM/IEEE series

The ACM/IEEE curriculum recommendation series define and update the “core of computing”, a description of the common knowledge base every computing graduate should be acquainted with. Besides, every sub discipline of computing defines core and elective content for sub-discipline specific curricula.

IT2017 (ACM / IEEE, 2017) describes this content in terms of performances (actions revealing understanding).

4.4.4.2 e-CF

e-CF is not an educational standard. Qualification is not discussed directly. It is discussed indirectly, in the definition of 23 iconic standardized professional profiles. These definitions refer to the European Qualification Framework. e-CF does not address the question “what should all computing professionals have in common”?

4.4.4.3 HBO-ICT 2009

The 2009 edition of the HBO-ICT framework emphasized specialization. Individual educational institutions were encouraged to design specialist curricula, and to clarify their scope by referring to (standard) professional tasks listed in the framework. Criteria for qualification should be deduced from the descriptions of these professional tasks. The question “what should all computing professionals have in common” is not addressed.

4.4.4.4 HBO-ICT 2014

The 2014 edition of the HBO-ICT framework emphasizes standardization. The methodology of defining an undergraduate curriculum and its qualification criteria is similar to the 2009 edition. But this report also recommends adding a theoretical foundation, common to all Bachelor-ICT programs, equivalent to 1 year or 25% of the curriculum.

4.4.4.5 HBO-Creative Technologies

The curriculum recommendations for Creative Technologies define four core competencies every program should cover. These are: technological, design, entrepreneurial and professional competencies. In order understand new technologies, graduates should be able to perform engineering type of practical research on Bachelor level. The Bachelor-’s level is defined in

Computing Curricula in Dutch Universities of Applied Sciences

generic terms, by referring to the policy agenda of the Union of Universities of Applied Sciences (HBO-Raad, 2009). We found no references to qualifications, specific to computing, to international frameworks, or to HBO-ICT.

4.4.5 KNOWLEDGE

In this section we will describe what the frameworks state about knowledge.

4.4.5.1 ACM/IEEE series

Knowledge is explicitly specified in the ACM/IEEE series. CS2013 and SE2014 (ACM / IEEE, 2013), (ACM/IEEE, 2014) distinguish 18 resp. 10 Knowledge Areas. We give an example: one of the CS2013 knowledge areas is Software Engineering, one of its core learning outcomes is “Describe how programming in the large differs from individual efforts with respect to understanding a large code base, code reading, understanding builds, and understanding context of changes” (CS2013, pag. 176).

IT2017 (ACM / IEEE, 2017) fits in the ACM/IEEE tradition. The curriculum description is competency-based; knowledge is one of the components of competency. We give two examples: The report (1) discusses mathematics education, and recommends including at least discrete structures in IT curricula; (2) it considers reflecting upon technology an essential performance, and mentions it frequently, as in: “Describe how the historical development of hardware and operating system computing platforms produced the computing operating systems we have today” (IT2017, page 95).

4.4.5.2 e-CF

e-Cf sees competences as a holistic concept, consisting of knowledge, skills and attitude. Descriptions of knowledge and skills are provided for each competence, in generic terms.

Operating systems are treated in section “Application Development” (Figure 4-2). We see the following description of knowledge: “K8 Operating systems and software platforms” and skills: “apply appropriate software and/or hardware architectures”. More than CS2013 or IT2017, e-CF seems to allow programs to focus on knowledge of current or emerging technology.

Computing Education in a Hybrid World

Plessius and Ravesteyn (Plessius & Ravesteyn, 2016) compared the e-CF knowledge areas with ACM's taxonomy of the domain of computing. They concluded: "e-CF more or less covers the IT domain, but some themes (such as 'theory of computing', 'mathematics of computing' and 'computing methodologies') appear only superficially on the e-CF". We agree with this observation. As we saw in section 4.2.2, e-CF relates competences to observable results. Most listed knowledge examples directly relate to practice, as in "K8 prototyping" (European Committee for Standardization, 2014, p. 25). Examples pointing towards deeper understanding, as in "K1 research methods, benchmarks and measurements methods" (p. 46), do occur, but scarcely.

4.4.5.3 HBO-ICT 2009

The HBO-ICT frameworks consider Bodies of Knowledge (BoK) the responsibility of the educational institutions. The 2009 framework recommends Institutions, not referring to a BoK, to define one. The professional tasks listed by the framework, and the professional products listed in the illustration, could serve as a starting point to draw up BoKs, specific for educational programs.

4.4.5.4 HBO-ICT 2014

The 2014 edition of the HBO-ICT framework states that the task-based description of the domain of ICT can be seen as the BoK of the domain. Different Bachelor of ICT programs will cover different parts of the description; the BoK is still seen as specific for programs and institutions. But the 2014 HBO-ICT framework also acknowledges the existence of a common BoK, and aims to give all graduates a solid theoretical foundation.

The content of this theoretical foundation is defined in terms of current tools, methods and techniques, as in: knowledge of widespread tools, testing methods, design methodology, modeling, architecture and business processes. Knowledge is deepened with specific components, depending on the profile of the program. These components are described in terms of professional tasks.

4.4.5.5 HBO-Creative Technologies

HBO-Creative Technologies' BoK consists of basics (common to all programs), visions and trends. Basic knowledge in technology is defined in terms of state-of-the art technologies. Visions are defined as "the most

Computing Curricula in Dutch Universities of Applied Sciences

important theories and methods from industry and academia”; trends refer to current and future developments. Visions and trends are not yet specified.

	Rationale	Learning objectives	Competencies	Educational Standards	Knowledge
ACM /IEEE	Development of the discipline Foster discipline oriented thinking	Support students' intellectual development Raise new generations of scholars Equip students to follow developments	Entrance level Master's degree. Provide industry competent workforce in a rapid changing world	Every graduate in a computing discipline should be acquainted with the common core of computing	Explicit description knowledge areas, learning objectives and indication of minimal amount of hours. IT2017: competency-based description include: knowledge
e-Cf3.0	Improve transparency of professional roles Explicitly meant as reference for computing education in Europe	Give students access to the pan-European labor market	Definition of knowledge and skills, required for standardized professional competence	Not a curriculum framework. Qualification is addressed indirectly, through the definition of 23 iconic professional profiles and corresponding levels of proficiency in e-competences	Indications provide in generic terms. Descriptions mainly in terms of current or emerging technologies. Theoretical themes under-represented

Computing Education in a Hybrid World

	Rationale	Learning objectives	Competencies	Educational Standards	Knowledge
HBO-ICT 2009	Provide a dynamic description of the domain of computing, compatible with intl. descriptions, to support r ecognizability of (new) curricular programs	Train students' skills and understanding needed to fulfill standardized professional tasks. Equip them to follow future developments in the field of computing.	Definition of standard professional tasks and levels of proficiency, illustrated by professional situations, which are typical for newly graduates.	None – framework emphasizes specialization. Institutions refer to standardized professional tasks to define specialized curricula.	No Body of Knowledge (BoK) provided. The domain description can be used by educational institutions to define a (specialistic) BoK.
HBO-ICT 2014	Provide a description of the domain of computing, to support recognizability of curricular programs in the Netherlands and across Europe.	Train students' skills and understanding needed to fulfill standardized professional tasks, equip graduates to learn and to contribute to the professionalization	Definition of standard professional tasks and levels of proficiency. Compatible with e-Cf 3.0	The framework recommends theoretical foundation, common to all graduates in computing disciplines.	BoK specific for educational institution. Common theoretical foundation described in terms of current tools, methods and techniques.
HBO- Creative Technologies	Boost innovation by supporting the top sector Creative Industries	Train students, able to adopt new (digital) technologies to develop groundbreaking people-oriented applications	Broad-based education and knowledge of new (digital) technologies.	Ability to perform engineering type of practical research .	(technological) BoK consists of basics, visions and trends. Basics defined in terms of state-of-the art technologies,

4.5 DISCUSSION

The five frameworks we have compared are very different, in their origins and in their aims. ACM and IEEE are international professional associations. The computing curriculum recommendations they jointly publish are mainly drawn up by scholars; industry and governmental organizations participate in the process. The e-CF project is funded by the European Union. e-CF is drawn up by industry; educational institutions participate as partners. HBO-I is an association of heads of departments offering undergraduate computing curricula. The HBO-ICT frameworks are drawn up by educational institutions; industry participates in the process. The framework for HBO-Creative technologies was written by heads of departments offering programs in Creative Technologies.

4.5.1 THE AIMS OF UNDERGRADUATE COMPUTING EDUCATION

We have found different ideas about the aims of undergraduate computing education. One view is: undergraduate computing education should ensure further development of the discipline. The other view is: computing education should train professionals able to apply state-of-the-art knowledge. In the first case, undergraduate curricula are seen as the first stage in academic education. Their aim is to develop the students' intellectual skills. Students, enrolled in such curricula, should understand the discipline's accomplishments and its wicked problems. Some of them will enter Master's, eventually PhD programs and will further develop of the discipline through research. Those who will not transfer to the next academic program will be hired by industry. The undergraduate curriculum can also be seen as a program, whose aim is to train highly skilled professionals needed by industry, in order to enable economic growth. Industry periodically needs professionals who fully understand the possibilities of the newest digital technologies. Such professionals should be able to translate their state-of-the-art knowledge into reliable digital artifacts. Some of them will pursue their education and enter Master's programs, eventually PhD. The educational system has many stakeholders; each of them will emphasize different aims.

Computing is a young discipline, and a dynamic one. The discipline is expanding: technology rapidly becomes obsolete, new professional roles emerge periodically. Curricula are limited. In curriculum design, long-term development and sustainability of training clash with the ambition to train state-of-the-art skills. Undergraduate computing education is facing a

Computing Education in a Hybrid World

wicked choice: should it focus at the intellectual development of its students, at the expense of the development of their skills or vice versa?

The international ACM/IEEE curriculum recommendations series points towards both directions. In 1989, the ACM/IEEE Task Force on the Core of Computing stated that undergraduate computing education has three main stakeholders: its students, industry and the discipline itself. Students should gain access to the Master's level and to the labor market; they should be well equipped to keep up with future developments. Industry needs skilled manpower. According to ACM/IEEE, the development of the discipline of computing goes hand in hand with the definition of educational programs for professional figures.

Although the European e-Competence framework is not a curriculum framework, it is explicitly meant as a guide for European educational institutions. It emphasizes the output of education, in terms of competencies, and delegates educational aims, like the students' intellectual development or the development of new disciplines, to educational institutions. These competencies occasionally include methodological issues, but most of them are described in terms of observable output, related to current technology. European institutions should keep this in mind while developing computing curricula. Focusing strictly on competencies requested by e-CF, that are related to current technology, will satisfy the industry's immediate need for qualified workforce. Yet, it is not likely to offer newly graduates overview of the discipline of computing, both in the large (understanding how different views on computing relate to each other) as in depth (reflection on technology). Such an approach is not likely to foster long-term development of the discipline, or to offer sustainable education to its students.

The Netherlands has a dual system. Academia focuses on research and development; it is responsible for the development of the discipline of computing. Universities of Applied Sciences, mainly prepare students to enter the labor market. Their computing programs focus on the application of state-of-the-art knowledge. Their graduates are not trained to develop the intellectual core of the discipline, but to contribute to the professionalization of the computing practice at hand. This duality might work in health care or hydraulic engineering, but seems problematic to us when it comes to computing. Human anatomy will not change over the years, nor will the principle of the communicating vessels. Educational health care programs, or programs for hydraulic engineers, offer future professionals understanding that will support them throughout their careers. This is less obvious in computing. Computing is evolving rapidly. A

Computing Curricula in Dutch Universities of Applied Sciences

system, built on the idea that knowledge development takes place in Academia and its application elsewhere, seems not adequate anymore. Industry, in particular the Creative Industries, might be ahead of Academia. The Dutch educational system seems not to take this possibility into account.

In the last two editions of the HBO-ICT framework, we have witnessed a shift in emphasis from specialization to standardization. One of the aims of the 2009 framework was to provide graduates with enough intellectual baggage to keep up with future professional developments. In 2014, the environment has changed. Europe has reached consensus about the profession of ICT. The 2014 edition of the framework reflects this consensus. The framework recommends a theoretical foundation common to all graduates, expressed in terms of knowledge of tools, methods for testing, design and modeling techniques. It emphasizes the role of computing practitioners in the professionalization of the computing practice. HBO-ICT increasingly calibrates its programs by referring to e-CF, but it also narrows its range. This could indicate a trend towards a limitation of the degree of freedom institutions take to define their own programs. In our opinion, increased standardization of programs towards e-CF entails the risk of inheriting the trade-offs we discussed above.

This applies even more to the Creative Technologies' hybrid curricula. They appear to be targeted to train professionals, able to use current and emerging (digital) technologies, without stating their relation to the field of computing. Hybrid curricula have limited time to address computing topics. Curriculum designers could feel pressured to focus on hands-on knowledge of the newest technologies, instead of basic understanding of computing, or understanding of the area of computing targeted by the program they are designing. In that case, we fear that these graduates' possibilities to build upon their knowledge of (digital) technology will be limited to few, less than 5, years after graduation.

4.5.2 HYBRID CURRICULA AND NEW PROFESSIONAL ROLES

The frameworks we have investigated adopt different strategies towards new professional roles. According to ACM/IEEE, educational institutions and industry share responsibilities in the standardization of computing curricula and the development of new ones. According to e-CF, the public authorities are responsible for the standardization of professional roles. Educational institutions bear responsibility for curriculum design. In the Netherlands, the situation is less clear. The domain of (applied) computing,

Computing Education in a Hybrid World

HBO-ICT, claims responsibility for the development of the existing professional roles, in coordination with Dutch industry. It increasingly refers to e-CF. The development of new curricula for the Creative Industries was stimulated by the Universities of Applied Sciences, with the definition of a new domain HBO-Creative Industries. It is unclear if this domain also assumes to share responsibility for the definition of the accompanying professional roles.

As a consequence of the differences described above, the position of new curricula differs too. IT2017 (ACM / IEEE, 2017), the last report in the ACM/IEEE series, provides guidelines for hybrid IT-curricula containing at least one year of computing related content. ACM/IEEE seems to include these hybrid curricula in their curriculum recommendations effort. e-CF lists “Digital Media Specialist” among the standardized European ICT profiles. e-CF standardizes this professional role, including it among the computing professions. In Dutch Universities of Applied Sciences, the trend is towards separation. A program training Digital Media Specialists was considered a possible variant of computing curricula in 2009. From 2014 on, the program belongs to a new domain called HBO-Creative Technologies, which appears to consider itself disjoint from ICT. HBO-ICT is entitled to professionalize the computing practice, but is cut off from at least some of the innovations in the digital area. This way, HBO-ICT will professionalize, but risks to jeopardize its evolution. HBO-Creative Technologies will boost innovation. However, the economic sector Creative Industries is likely to suffer from growing pains if its companies will not share in the know-how of HBO-ICT, or in the future professionalization of the computing practice. This approach seems to overlook possibly harmful long-term effects for Dutch economy.

4.6 CONCLUSIONS AND RECOMMENDATIONS

The ACM/IEEE curriculum recommendations program strives at defining and monitoring the intellectual core of the discipline of computing. But computing is also an applied discipline. The last report of the series acknowledges this and is structured around “performances” instead of “knowledge”. In line with its academic approach, the report considers “Reflection” upon technology a core performance.

The European e-CF is not a curricular framework. It mainly focuses on the employers’ requirements. Other stakeholders’ requirements as the students’, or society’s seem in danger of being overlooked. European

Computing Curricula in Dutch Universities of Applied Sciences

educational institutions offering computing programs should cope with this issue.

The Netherlands has a dual system, with academic and applied undergraduate computing curricula. Academia is responsible for the development of the discipline, the applied track for its professionalization. We fear that this separation will not be fruitful in the long term for the rapidly evolving discipline of computing.

The last report of the ACM/IEEE series includes examples of hybrid curricula. E-Cf lists at least one profession among the standardized European ICT professional profiles that is considered hybrid in the Netherlands. But in the Netherlands, applied computing curricula seem to direct towards separation from hybrid curricula. The Dutch domain of applied computing seems to narrow its range, while hybrid domains emerge. We fear that this separation will prevent computing to professionalize in innovative ways. As for the new domain of Creative Technologies, we recommend it to reconsider its relation to computing, to explicate its scope and implement its choices in the final draft of its curriculum framework.

Dutch applied computing curricula calibrate their theoretical base by referring to current technologies. Curricula in the Creative Technologies refer to emerging technologies. Extensive learning skills should ensure these graduates' long-term employability. We doubt that "extensive learning skills", not supported by overview of the discipline, will be sufficient to offer their graduates a sustainable professional perspective. We fear a loss of investments for society. Understanding computing fundamentals, or understanding the *raison d'être* of current technology (besides learning how to use it) seems a better approach to us. We recommend the HBO computing disciplines, including the hybrid ones, to approach technology as illustration of general computing principles, rather than focusing on lecturing technology itself.

Academic and professional education are separated in the Netherlands. Academia is responsible for the development of the discipline of computing. HBO-ICT sets the standards for its professionalization; a process Academia is not involved in. Innovation and the design of ground-breaking applications involving (digital) technology is delegated to a new HBO-domain, Creative Technologies, that seems to neglect its relation with computing. We doubt that a too strict separation between Academia and practice can be fruitful in this area. Computing is a rapidly evolving discipline, with an important intellectual component. We recommend HBO-ICT and Creative Technologies to intensify their collaboration with each

Computing Education in a Hybrid World

other and with Academia, to (1) define a sustainable theoretical base for their graduates, (2) allow the creative industry to benefit from the professionalization of the computing practice and (3) prevent computing from being cut off from evolution. We recommend Dutch government to support these collaborations.

PART I - Conclusions

In this Part, we have reflected upon the discipline of computing and upon (Dutch) national and international curricular frameworks for undergraduate education. Our aim was to identify the aims and rationale of undergraduate computing curricula, both (Dutch) national and international. The purpose was to prepare for the formulation of recommendations for designers of undergraduate computing curricula in general and hybrid computing curricula specifically.

RESEARCH QUESTION RQ1

RQ1 What are possible approaches to computing and computing education?

We have found 3 approaches to computing (or cultural styles), all fundamental for computing: theory, science and engineering. The theoretical cultural style describes abstract structures in an unambiguous way. The scientific cultural style addresses the question: do our models match with the world they describe? The engineering cultural style investigates how to design and implement reliable systems. According to Tedre and Apiola (Tedre & Apiola, 2013), these cultural styles embody different epistemological values, scarcely compatible with each other.

Tedre and Apiola reflect upon the role of cultural styles in education. They recommend matching cultural style with learning objectives, in order to design efficient education. They also recommend that all computing educators fully understand the complex epistemological background of the discipline, and warn for bias introduced by hidden ethos in elevating one of the cultural styles above the others.

The ACM/IEEE Task Force on the Core of Computing (Denning, et al., 1989) acknowledges the existence of three major paradigms or cultural styles of computing, and states that they all are fundamental for the discipline. The ACM/IEEE curriculum recommendations are based on this standpoint.

e-CF (European Committee for Standardization, 2014) does not state how computing should be approached. The framework is competence based. It defines competencies as “a demonstrated ability to apply knowledge, skills and attitude for achieving observable results” (European Committee for Standardization, 2014, p. 5). Knowledge is fundamental in this framework, not the paradigms behind that knowledge, or the interplay between these paradigms.

The Dutch HBO-ICT framework (Valkenburg, et al., 2014) too is competence-based. We found no references to possibly different approaches to

Computing Education in a Hybrid World

computing in the framework documentation. We have expressed the fear that a too strict competence-based approach in an applied computing curriculum, not addressing reflection upon the discipline, could have unwanted side effects. It might result in training professionals prone to overlook other possible approaches to computing than the one, preferred by the sub discipline they were trained in.

We saw examples of hybrid curricula, and argued that these curricula can be typed by explicating the cultural style or mix of cultural styles they adopt while approaching computing.

RECOMMENDATIONS

We recommend that all educators of computing topics understand the complex nature of computing. This applies to hybrid curricula and to Dutch undergraduate applied curricula in particular.

We recommend HBO-I to stress the importance of “reflection upon (the evolution of) the discipline” in the next version of the HBO-ICT framework.

We recommend designers of hybrid curricula to refer to the three cultures of computing to type the curricula they design. It is inevitable for designers of hybrid curricula to make choices: which part of computing should be included, and why? A characterization of the computing-related part of a curriculum by referring to its orientation gives insight in the related trade-offs.

RESEARCH QUESTIONS RQ2, RQ3

RQ2 What are the aims of undergraduate computing education?

We have found different ideas about the aims of undergraduate computing education. One view is, that undergraduate computing education should ensure further development of the discipline. The other view is: computing education should train professionals able to apply state-of-the-art knowledge. Different stakeholders of the educational system will emphasize different aims: Academia will ask for intellectual development, Industry will ask skilled workforce, students will ask education offering them a sustainable view on the discipline.

Computing is a young discipline, and a dynamic one. The discipline is expanding. Technology rapidly becomes obsolete, new professional roles emerge periodically. Curricula are limited in time. Undergraduate computing education is facing a wicked choice: should it focus at the intellectual development of its students, possibly at the expense of the development of their ready-to-use skills? Vice versa, should it focus at

training state-of-the-art skills, possibly at the expense of their intellectual development?

The international ACM/IEEE curriculum recommendations series points towards both directions. In 1989, the ACM/IEEE Task Force on the Core of Computing stated that undergraduate computing education has three main stakeholders: its students, industry and the discipline itself. Students should gain access to the Master's level and to the labor market; they should be well equipped to keep up with future developments. Industry needs skilled manpower. According to ACM/IEEE, the development of the discipline of computing goes hand in hand with the definition of educational programs for professional figures.

Although the European e-Competence framework is not a curriculum framework, it is explicitly meant as a guide for European educational institutions. It emphasizes the output of education, in terms of competencies. These European e-competences are described by stating their name, giving a generic description, indicating which tasks are related to different levels of proficiency, and listing examples of related knowledge and skills (not exhaustively). Descriptions of competencies occasionally include methodological issues, but most of them are described in terms of observable output.

The Netherlands has a dual system. Academia focuses on research and development; it is responsible for the development of the discipline of computing. Universities of Applied Sciences, mainly prepare students to enter the labor market. Their computing programs focus on the application of state-of-the-art knowledge. Their graduates are not trained to develop the discipline but to contribute to the professionalization of the computing practice at hand.

RQ3 What is the purpose of undergraduate computing curriculum recommendations series, i.e. of (RQ3a) international curriculum recommendations series and (RQ3b) curriculum recommendations series for Dutch Universities of Applied Sciences?

RQ3a The purpose of undergraduate computing curriculum recommendation series: international series.

We investigated the ACM/IEEE curriculum recommendations and the European e-Competence framework e-CF.

ACM/IEEE: The goal of education in general is to gain competence in a domain. The domain of computing is complex, because computing is a rapidly evolving intellectual discipline that can be approached from

Computing Education in a Hybrid World

different points of view. The joint ACM/IEEE committees identify various sub-disciplines of computing, all (1) having substantial theoretical component (2) handling significant abstractions (3) dealing with important design and implementation issues and (4) that are sustained by one or more research communities providing their own literature (Denning, et al., 1989). Research communities of these sub disciplines validate their assumptions by investigating them from different points of view: theoretically, scientific and from an engineering point of view. A considerable part of all undergraduate curricula is devoted to the “core of computing”. This is a Body of Knowledge, common to all sub disciplines, each graduate in computing should master. It illustrates the three possible approaches to computing: formal, scientific and engineering. With the curriculum recommendations series, ACM/IEEE aims to foster discipline-oriented thinking.

e-CF: e-CF is not a curriculum framework, but is intended as an industry standard. The framework emphasizes the output of education. It describes competencies, associated to standardized professional roles. The European Union claims responsibility for definition and description of (standardized) professional roles in terms of observable application of knowledge and skills. It delegates the development of curricula educating these professionals to educational institutions. e-CF aims at preventing shortage of qualified ICT manpower, by establishing a common language to express professional roles and competencies across Europe.

RQ3b The purpose of undergraduate computing curriculum recommendation series: series for Dutch Universities of Applied Sciences

The Dutch Universities of Applied Sciences (HBO) train practitioners to apply knowledge that was developed in Academia. The HBO-ICT frameworks' principal aim is to support designers of curricula in the specification of programs for the education of computing practitioners, needed by the Dutch (and pan-European) labor market. This also applies to the new Dutch framework for hybrid curricula we have viewed, HBO-Creative Technologies.

The last Dutch framework for computing curricula at the Universities of Applied Sciences (HBO-ICT 2014) (a) trains computing professionals for the pan-European labor market, (b) having a common theoretical foundation, described in terms of state-of-the art technology (c) having extensive learning skills (d) able contribute to the professionalization of their profession.

The question rises, if the “theoretical foundation”, common to all HBO-ICT graduates, should include understanding of three cultural styles of computing. In my opinion, understanding the three cultural styles of computing and their different approaches to research is necessary to those, who want to contribute to the development of a unified discipline of computing. This includes educators of computing related topics. For practicing professionals, a balanced knowledge base seems more important than ability to switch between epistemological points of view. ACM/IEEE have described such a knowledge base, illustrating the full spectrum of computing. But for practicing professionals it is important to be able to keep up with future development. How this ambition relates to the epistemological differences we have found in the three cultural styles of computing, is a question that ought to be addressed.

The Dutch framework for curricula in the Creative Technologies (HBO-Creative Technology, hybrid) aims at boosting innovation by supporting the Dutch economic top sector Creative Industries. It does not state its relation to computing explicitly.

RECOMMENDATIONS

e-CF describes the output of the educational system. Other stakeholders’ requirements, as the students’ need for a sustainable career perspective, or the discipline’s long-term perspectives, seem in danger of being overlooked. European educational institutions offering computing programs should cope with this issue. This applies to Dutch institutions, offering applied computing programs in particular.

We recommend national and international organizations to explore the question, of how differences in the research values of the three cultural styles of computing relate to the education of computing practitioners.

We recommend HBO-I to define a sustainable theoretical base for its curricula in collaboration with Academia. We recommend Dutch government to support the collaboration between institutions offering computing curricula in Academia and in the applied domain.

We recommend designers of hybrid computing curricula to acknowledge that their programs are related to computing. We recommend them to take responsibility for the definition of new professions, and to participate in their development, in collaboration with related computing disciplines, Academia, the industry and the public authorities.

Computing Education in a Hybrid World

RESEARCH QUESTIONS RQ6, RQ7

RQ6 Which subject-specific strategies were recommended in the past?

RQ7 Which subject-specific strategies can we recommend?

The first Task Force on the Core of Computing (Denning, et al., 1989) recommended using differences between programming languages as a vehicle to discuss differences between approaches to computing.

In order to support graduates in keeping up with the changes in the domain of computing, the first Task Force on the Core of Computing recommended an inquiry-based approach to education. Rather than lectures presenting answers, the Task Force recommended acquaintance with the computing literature and with the related research methods. But according to Tedre and Apiola, the three cultural styles have fundamentally different approaches to research.

Up-to-date knowledge of one of the sub disciplines of computing and acquaintance with the related research methods is more likely to point towards a specialist career path than towards overview of the discipline.

In general, we doubt that an inquiry-based approach in the undergraduate curriculum can be combined both with the Task Forces' aim to foster discipline-oriented thinking and its goal to train students to access the labor market.

Tedre and Apiola recommend aligning learning objectives with the cultural style because a mix would not result in successful educational interventions.

RECOMMENDATIONS:

The combination of the first ACM/IEEE Task Forces' aim to foster discipline oriented thinking, the curricular goal to train students for entering the labor market, and the recommended instructional strategy (inquiry-based) ought to be reconsidered.

At course level, we endorse Tedre and Apiola's recommendation to align the learning objectives with the cultural style, but we also recommend to cultivate awareness for differences in cultural style in computing curricula.

We endorse the first ACM/IEEE Task Force's recommendation to use programming languages, or programming paradigms, as a vehicle to discuss differences between approaches to computing. In general, we recommend HBO-ICT to address the problems behind the development of technologies rather than focusing on knowledge of state-of-the-art technology.

PART I - Conclusions

Approaching technology as an illustration of more general principles will support the graduates' future understanding of the discipline.

As for hybrid computing curricula, we recommend that lecturers of computing topics (1) should have a CS degree or equivalent, (2) discuss aims and boundaries of hybrid programs with their students and (3) refer to technology as a vehicle for that discussion.

Part II—How Do Students Understand The Subject

RESEARCH QUESTIONS RQ4, RQ6, RQ7

RQ4 Do students, who were educated in different computing disciplines, develop the same mental models for the abstract concepts they work with? I.e., are different approaches to computing interchangeable?

RQ6 Which subject-specific strategies were recommended in the past?

RQ7 Which subject-specific strategies can we recommend?

After a literature review on cognitive aspects of programming and reasoning, we will describe the results of an experiment we conducted in a Dutch University of Applied Sciences (HBO-ICT). We recruited senior students, enrolled in different undergraduate computing programs. They were about to complete their classes and start their internships. We investigated if they conceptualize abstract entities in the same way.

All these students had enrolled between 2009 and 2013 in computing programs of the Utrecht University of Applied Sciences. Some of them followed a program in Business IT and Management, others in Software Engineering. Figure II-1 describes the emphasis of Business IT and Management in terms of (1) software lifecycle activities (columns: Analysis, Advice, Design, Implementation and Maintenance), (2) architectural layers (rows: User Interaction, Business Processes, Software, Infrastructure and Hardware Interfacing) and (3) the levels of proficiency newly graduates should attain (in the cells, values: 1..3).

The Business IT and Management curriculum emphasized (and still emphasizes) Business Processes. As we can see in Figure II-1, the dimension ‘Software development’ is considered secondary.

Computing Education in a Hybrid World

	analyse	advies	ontwerp	realisatie	beheer
Gebruikersinteractie					
Bedrijfsprocessen	3	3	3	3	3
Software	2	2	2	1	1
Infrastructuur					
Hardware interfacing					

Tabel 2-1 Beheersingsniveaus Business IT & Management

Figure II 1 Levels of proficiency BIM (Hogeschool Utrecht, 2012)

Students enrolled in Computer Science can major in different programs. Software Engineering is one of them. Its curriculum focuses on the architectural layer Software (Hogeschool Utrecht, 2012), how to analyze, design and develop it.

	Analyseren	Adviseren	Ontwerpen	Realiseren	Beheren
Gebruikersinteractie	1	1	1	1	
Bedrijfsprocessen	1		1	1	
Software	3	3	3	3	1
Infrastructuur		2			
Hardware interfacing					

Tabel 2-2: Beheersingsniveaus Informatica Software Engineering

Figure II 2 Levels of proficiency SE (Hogeschool Utrecht, 2012)

In 2013-2014, Information Engineering was a major program of Computer Science at the Utrecht University of Applied Sciences. The curriculum shared its attention between the architectural layers User Interaction, Business Processes and Software (Figure II-3). The emphasis was on design and development of software that is meant to support user interaction, in particular of User Interfaces.

	Analyseren	Adviseren	Ontwerpen	Realiseren	Beheren
Gebruikersinteractie	1		2	2	2
Bedrijfsprocessen	2	1	1	1	1
Software	2	2	2	3	1
Infrastructuur					
Hardware interfacing					

Tabel 2-3 Beheersingsniveaus Informatica Information Engineering

Figure II 3 Levels of proficiency IE (Hogeschool Utrecht, 2012)

Part II—How Do Students Understand The Subject

In this Part, you will find two papers. The first one describes work in progress. In line with the considerations in chapter 2, we had hypothesized differences in mental models for the notion of “object” of students Software Engineering and students Business IT and Management. We did find differences, but not the differences we had expected. In the second paper, we further analyzed our results to better understand the differences we had found across the groups.

We will return to the unexpected results of this experiment in chapter 10.3.

5 Cognitive Aspects of Software Development³

Keywords: POP-I.A. group characteristics, POP-II.A. individual differences POP-II.C. working practices, POP-V.A. mental models

ABSTRACT

Computer Science has evolved towards a discipline with different branches. Scholars study and define artefacts from different viewpoints: computer languages, environments, paradigms. Practitioners work with these artefacts. They design, produce and link software that was designed according to different paradigms. They often work in multidisciplinary teams. We are interested in the communication between these practitioners. Do they refer to the same concepts when they use the same words? We designed an experiment to assess this.

5.1 INTRODUCTION

In the past couple of decades, Computer Science has developed different branches, each with its own body of knowledge and its own problem area. Practitioners operate in the real world; the problem areas they encounter often are heterogeneous. Choosing an approach for the solution of problems, or combining results from different branches of Computer Science, is not at all straightforward.

Object-oriented software and relational databases have their origins in different approaches to the digitalization of information. There are evident differences between the Object Oriented and the Relational paradigm. The semantics of the UML is not specified in a formal way, whereas the semantics of the relational model is given in terms of mathematical concepts. In database theory, data models define database universes for databases that are built to last. OO models describe program structures at a certain moment in time. They are designed to be adapted to changing circumstances during their lifecycle. Ireland et al. (Ireland, 2011) remark that

³ Based on: L. Benvenuti, G.C.van der Veer: *"The Object-Relational impedance mismatch form a cognitive point of view"*, In: B. du Boulay and J. Good (eds) Psychology of Programming Interest Group annual conference 2014, 25-27th June, Brighton, United Kingdom

Computing Education in a Hybrid World

the concepts underpinning the relational model are prescriptive and formal, while those underpinning object schemas are more descriptive. Finally, a database universe defines a closed world. Each row corresponds to a true statement about that world, and conversely, a precise meaning is attributed to the absence of rows: objects that are not represented in the database world are assumed not to exist in the problem domain. That conclusion is less straightforward in Object Oriented environment.

In Computer Science departments, different groups of scholars use different kinds of language, depending on the paradigm they study. SQL, the query language for relational databases, is a Domain Specific Language, while Object Oriented languages such as Java are considered General Purpose Languages. In the imperative-declarative spectrum, SQL is seen as a declarative language while OO languages such as Java and Smalltalk are held to be more imperative (van Roy, 2008).

What happens when the paradigms meet, when engineers link together software that was designed from different points of view? This area is characterized by a persistent problem, the Object/Relational impedance mismatch. Many attempts have been made to solve the Object/Relational impedance mismatch, mainly by developing new software solutions: Object Relational Mappings. These mappings are based on the assumption that relational data models can be interpreted as class diagrams, where tables correspond to classes and rows correspond to objects. This is problematic, though. In an OO program, an object has an identity independent of its state, but specific to the program execution. Various objects with the same state can coexist during one program execution. In the relational model, the row is identified by its attribute values (the state of the row) and it is accessible through set operations only: not directly. Direct access to objects during the execution of OO programs is possible and is based on navigation. The conclusion is that despite the supposed correspondence between rows and objects, these mappings are problematic. The Object/Relational impedance mismatch has been postponed but has not been resolved yet. (Ireland, 2011) (Zicari, 2012)

The problem with these solutions is their validation. Multidisciplinary teams that work on tangible artifacts have a common field of application: the world we live in. Engineers from different backgrounds can discuss advantages and disadvantages of machine control software, plant engineering software or computer graphics software, knowing that in the end they refer to the same tangible or visible artifacts. Computer scientists, in particular database engineers, do not work on tangible artifacts: they work on abstract entities. Technical solutions of the Object Relational

Mismatch can only be evaluated if working practitioners perceive a common field of application, if they work with the same abstract entities. The question here is, whether they do.

In this paper, the emphasis is on human characteristics, rather than on the formal properties of programming and modeling languages. We will focus on mental representations of abstract entities involved in programming and modeling. In section 5.2 we explore the notion of a mental model. In section 5.3 we give an overview of research on cognitive aspects of programming and database interaction, followed by reflections in section 5.4 in section 5.5 we give the outline of a first experiment we designed to assess possible differences in mental models between (OO) programmers and database professionals, followed by preliminary results. We will draw preliminary conclusions in section 5.6.

5.2 BACKGROUND

5.2.1 MENTAL MODELS

One of the first to identify the concept of mental model was K. Craik, (Johnson-Laird, *Mental Models*, 1989), (van der Veer & Puerta Melguizo, *Mental Models*, 2002) who suggested that the mind constructs “small-scale models” of reality and uses them to anticipate events. Ph. Johnson-Laird elaborated the concept further and formulated a theory of mental models, meant to explain human thinking and reasoning. According to Johnson-Laird, people do not only apply inference rules while reasoning; they also consider the semantic content of the problems they are solving. While reasoning, people construct mental models representing the (semantic) information of the problems they are considering. A mental model or mental representation does not provide a complete description of a problem situation, but a simplification. Mental representations are based on pre-existing knowledge plus new, problem-specific, information and actual needs.

An influential article by D. Norman (Norman, 1983) is more specifically concerned with mental models in Human-Computer Interaction. Users, states Norman, construct mental models of computer systems incrementally, while interacting with these target systems. This process results in models that are constrained by the users’ prior knowledge, needs and context. Most of the times, these models are incomplete. They are limited by the human information processing system, by experience and by needs that can be contrasting: the need to focus and the need to retain

Computing Education in a Hybrid World

important details. They are parsimonious: people often prefer adding actions in order to reduce mental complexity. User mental models are unscientific and unstable. They evolve over time: people learn, people forget. People happily use metaphors to simplify their models. Nevertheless, mental models are functional to support various tasks such as planning, execution, assessment of results and understanding of unexpected events.

Despite the attention to mental models and their conception, there is little agreement on the exact definition of the term “mental model”. Does the term refer to temporary structures in Working Memory (WM) or knowledge structures in Long Term Memory (LTM)? Cañas and Antolí (Cañas, 1998) introduce this definition: a mental model is *“the dynamic representation that is formed in WM combining the information stored in LTM and the extracted information from the environment”*.

Knowledge, in particular practitioners’ knowledge concerning computing paradigms, is acquired by education and experience and stored in LTM. Johnson Laird describes “mental models” as knowledge chunks in LTM that evolve in time. According to Norman, evolution is not necessarily positive: improvement is possible, deterioration also. These knowledge chunks can only be used if they are retrieved and instantiated in WM, in what we will call a mental model from now on, following Cañas’ and Antolí’s definition. The function of these mental models is to represent relevant aspects of reality in WM, and this includes the representation of actions: mental models are partially runnable.

Human WM is limited to a small number of chunks (7 ± 2 chunks) (Baddeley, Eysenck, & Anderson, 2009), for brief periods of time after each chunk has been given attention (ca. 30 sec). Users, practitioners and human beings in general, take these limitations into account. They adopt strategies to keep mental models manageable in WM. The question here is: can we assume mental models of practitioners with different backgrounds to be compatible with each other?

5.2.2 INDIVIDUAL PREFERENCES

Schwank’s research (Schwank, Cognitive structures and cognitive strategies in algorithmic thinking, 1993) (Schwank, Zur Konzeption prädikativer versus funktionaler kognitiver Strukturen und ihrer Anwendung, 1996) concerns mainly the didactics of mathematics. She found differences in the way people select mental models to represent knowledge. Schwank distinguishes a predicative and a functional approach to mathematic thinking. Predicative thinking is “static grasping”, thinking in terms of

Cognitive Aspects of Software Development

judgments and relations; functional thinking is “dynamic grasping”, thinking in terms of actions, processes and their effects. Schwank adopts this metaphor: predicative thinking is like solving puzzles, where one looks for (static) patterns. Functional thinking is mechanical thinking, like using gear: the focus is on what happens next.

Most mathematical problems can be solved either way; this also applies to many problems in computer science. To explain the working of a given algorithm, one of Schwank’s subjects wrote: “the program always counts one (...) more than the triple of the content of (...) R1”. This describes the relation between input (the content of R1) and result. According to Schwank, this indicates a preference for predicative thinking. Another respondent characterized the same algorithm with: “as long as something is in R1, take it off and multiply it by three. Then, at the end, add one”. Here, the result is derived from the input, and the algorithm’s description highlights the process. Schwank calls this functional thinking.

Functional thinking matches the simulation of mechanical processes. Schwank cites as example the Turing machine, an idealized model of a computer. To illustrate predicative thinking, emphasizing judgements, she refers to set theory (Schwank, Cognitive structures and cognitive strategies in algorithmic thinking, 1993) (Raven, 1965, cited by Schwank)

To validate her theory, Schwank used Raven’s progressive matrices (Raven, 1965 cited by (Schwank, Analysis of eye-movements during functional versus predicative problem solving, 2002)). 3x3 matrices lacking the lower right symbol were presented to the subjects, which were asked to construct the last symbol to complete the sequence (instead of choosing the last symbol as is requested in Raven’s intelligence test). The subjects were asked to argue why they draw that particular symbol. Schwank found support for the thesis that the preference for a predicative or functional mode in problem solving is a stable individual characteristic.

Schwank (Schwank, Cognitive structures and cognitive strategies in algorithmic thinking, 1993) compares her findings with other cognitive theories. In particular, she wonders if the distinction she makes matches with the distinction between declarative and procedural knowledge, as elaborated by Anderson (as cited by Schwank). The conclusion is negative: *declarative / procedural* applies to the kind of knowledge as stored in Long Term Memory, while *predicative /functional* is a “property of the structure of knowledge representation” (Schwank, Cognitive structures and cognitive strategies in algorithmic thinking, 1993). Schwank illustrates this with the mathematical operation of division. Division can be approached functionally and predicatively. Both approaches generate components (knowledge) in

Computing Education in a Hybrid World

what Schwank labels “production memory” and components in declarative memory, but different components. In other words: *predicative/functional* applies to the way information is processed and encoded (Schwank, Cognitive structures and cognitive strategies in algorithmic thinking, 1993). Schwank remarks that preference for one of the approaches is likely to have consequences for the knowledge that is stored in Long Term Memory.

About Johnson-Laird’s theory of mental models, Schwank concludes that Johnson-Laird “*neglects to include an independent functional component in his theory of mental models*” (Schwank, Cognitive structures and cognitive strategies in algorithmic thinking, 1993).

Schwank’s distinction concerns problem-solving skills. No assumptions are made as to the nature of mathematical knowledge. One could argue that some problems seem to request a functional approach, while others ask for a predicative approach. Schwank’s conclusion is that, even with problems that seem to ask for a declarative approach, functional thinkers will adopt a functional strategy. Sometimes, they will perform worse than when solving problems that are more akin to be approached functionally; they might need more time to solve the problem, they may make more mistakes than predicative thinkers and vice versa.

Schwank aims to introduce the functional dimension in the discourse on the didactics of mathematics. She uses the distinction between predicative/functional thinking to better understand gender differences in learning mathematics, and cultural differences (between German students, Indonesian students and Chinese students). For our paper, the distinction is relevant because it might explain differences in mental models adopted by groups of computing practitioners differing in the paradigms they use and are educated in.

5.2.3 ASSESSING MENTAL MODELS: THE TEACH-BACK PROTOCOL

Mental models cannot be observed directly. Many authors draw conclusions about a respondent’s mental models based on their behavior (Moray, 1998). Van der Veer adopts a different strategy: he asks the respondents. Van der Veer extended an hermeneutic method designed by G. Pask, intended to elicit information about mental models (the Teach-Back method), and adopted it to elicit mental models from users interacting with computers and to detect differences in mental representations (van der Veer, Learning, individual differences and design recommendations, 1990).

A situation is simulated where the respondent has to interact with a computer. The respondent is asked to explain the computer’s functioning to

Cognitive Aspects of Software Development

an imaginary counterpart, a colleague or a student, who has similar experience with the situation. Questions are presented on white sheets of paper, and the respondents are instructed to express themselves in whatever way they consider most adequate: text, drawings, keywords, diagrams etc. In this manner, the subjects are encouraged to externalize the mental model they made of the situation. Next, the protocols are scored along pre-defined scoring categories in order to map the respondent's mental representations. Rating implies (1) reading the protocol in its entirety and trying to understand fully what it says; (2) trying to formulate how the subject represents the space of the teach-back question and (3) classify the responses into relevant categories for the purpose of the study. Rating is done independently by two persons in order to safeguard reliability.

5.3 LITERATURE REVIEW

In the next sections, we will review the literature on mental representations of programming constructs.

5.3.1 COGNITIVE ASPECTS OF (OO) PROGRAMMING

The academic debate on the cognitive process supporting programming was especially active from the 1970s to the late 1980s, when the dominant question became: "how do programmers make sense of code?". Later, the emphasis switched to the relationship between procedural and Object Oriented (OO) programming.

Robins, Rountree and Rountree (Robins, Rountree, & Rountree, 2003) provide us with an extensive literature review on research concerning learning and teaching programming between 1970-2003, in order to formulate recommendations for teachers. In a survey study, D tienne (Detienne, 1997) reviews empirical research on OO design and assesses claims about the cognitive benefits of the OO paradigm. We will focus on three topics: mental models involved in learning programming, strategies of program creation, and comparison between the procedural and the OO paradigm.

Robins et al. mention different kinds of mental models involved in learning programming: mental models of the "task, problem or specification" that has to be solved by the program, mental model of the programming language, and mental models of the behavior of the running program. Many studies have noted a central role played by a model (or abstraction) of the computer. Du Boulay et al. (du Boulay, O'Shea, & Monk,

Computing Education in a Hybrid World

1989) call this the “notional machine”, an idealized, conceptual computer that is defined with respect to the language. Novices should develop an appropriate notional machine to master a programming language: the notional machine underlying Pascal is very different from the machine underlying PROLOG. A study by Mayer (Mayer, 1989) confirms that students supplied with a notional machine model perform better while solving some problems than students who are not given the model (Robins, Rountree, & Rountree, 2003).

A model of program comprehension was provided by Pennington (Pennington, 1987). Comprehension is seen as the assignment of meaning and occurs in the context of a problem domain. Attribution of meaning starts from a text (the language specific code in the programming domain) which is re-organized in mental representations with help of available knowledge structures. Pennington describes Text Structure and Plan Knowledge. Text Structure knowledge emphasizes the role of abstract knowledge of text structures. Plan Knowledge emphasizes the role of content-dependent knowledge. Abstract plans are achieved by specific program functions. Plan Knowledge tells us which plans are achieved by which functions.. Pennington investigates which structure is dominant in the cognitive organization needed for program comprehension. Pennington studies expert FORTRAN and COBOL programmers, and finds significant differences in their performances of comprehension tasks. She observes that one of the possible explanations is that the structure of the language influences the mental representation of programs.

Détienne also mentions mental simulation. She points out that the strategy of mental simulation was documented by authors of several studies on procedural design. Few studies were conducted on this subject about OO design; they concerned small groups of programmers and the findings are contradictory.

One of the central notions when describing programming knowledge is the “plan”, understood as a structure or schema in Long Term Memory. According to Rist (Rist, 1995) “the plan is the basic cognitive chunk used in program design”, but Rist also observes that the exact meaning of “plan” varies between authors. Rist also refers to the term “script”. Détienne uses the term “schema” to indicate knowledge chunks. Robins et al. observe that the term “plan” or “schema” is ill-defined. In general, the term “script” indicates a type of schema relating to the typical sequence of events in common situations. “Plan” is used in relation to problem-solving activities. People use scripts to keep track of sequences they have experienced, but plan ahead. Considering that our investigation concerns mental models as

Cognitive Aspects of Software Development

instantiated in WM (though based on knowledge in LTM), we will avoid using these terms.

Détienne (Detienne, 1997) describes three strategies guiding OO design activities. Two of these (function-centered and object-centered) are declarative strategies, where static characteristics guide solution development, and one is a procedural strategy (procedure-centered), where dynamic characteristics prevail. Rist (1996, as cited by Détienne) describes similar strategies (based on roles, objects and goals). Each of these studies shows dominant strategies for groups of designers, but they also show that individual designers switch between strategies while solving problems.

Independently of each other, Détienne and Rist investigate the conditions which trigger the use of one strategy rather than another. Their results do not fully match. In separate studies, Détienne finds (1) that novices tend to adopt a procedural strategy, where experienced designers more often choose an object-centered approach, and (2), in a study conducted with Chatel, that expert OO designers match their strategy to the problem. A related issue concerns the classification of problems for OO design. Many distinctions have been proposed. In particular, the procedural-declarative distinction seems not to influence the design activity (Detienne, 1997).

Rist concludes that strategies of experienced OO designers are mainly procedure-centered, where Détienne reports that the procedure-centered strategy is most commonly used by novices. Détienne explains this by noting that different authors categorize the subjects in different ways. Rist's experienced designers were students, and might have been closer to Détienne's novices than to the (professional) experts she observed.

OO programming languages were expected to improve understandability, in comparison to traditional languages. More precisely, the OO approach has claimed to make modeling the problem domain easier for programmers. Studies by Wiedenbeck et al. (Wiedenbeck, Ramalingam, Sarasamma, & Corritore, 1999) compared novices' comprehension of procedural and OO programs. Wiedenbeck et al. found that for short programs, there was no significant difference between languages, but if multiple classes were used, procedural programmers did better. The authors conclude that the distributed nature of OO programs may make it more difficult for novices to form a complete mental representation of an OO program than of a procedural program. They also suggest that OO novices focus on program model information, instead of focusing on the problem domain. Robins, Rountree and Rountree conclude that there is little

Computing Education in a Hybrid World

support for the claim that the OO paradigm focuses the programmer on the problem domain, especially where novice programmers are concerned.

Détienne investigates the OO claim of naturalness and greater ease of design. Novice programmers have difficulties in class creation and in articulating procedural and declarative aspects of their solutions. They need to start with a procedural representation of the situation. This seems to indicate that knowledge is organized in terms of procedures, not in terms of objects and relations. But for expert OO designers, the claims find support. Experts do analyze problems in terms of objects and their relationships. Also the claim that OO designers design solutions that are closer to the problem domain is supported. Expert OO designers seem to shift between object view and procedure view. This may support Rist's claim, that OO programming is not different from procedural programming, "it is more" in that it adds a class structure to a procedural system (Rist, 1995).

Détienne comments on the oddity of the situation: if OO design is driven by domain knowledge, then the biggest benefit should be observed in novices; but this is not the case. In her explanation, she notes that at the time no studies were conducted on native OO designers. It would be interesting to compare novice programmers learning to design in the OO and the procedural paradigms. This is what Wiedenbeck et. al. did, but their conclusions match Détienne's previous findings.

5.3.2 COGNITIVE ASPECTS OF USER-DATABASE INTERACTION

Today, the Relational Model is the leading model in database theory. Its main feature is that the model facilitates re-use of data. Once collected and stored, data can be used for other purposes than it was originally intended for. There is no need to know how data are stored in memory in order to write a query: knowledge of the (abstract) data model of the database is sufficient. The Relational DataBase Management System (RDBMS) provides the interface between the user writing the query and the physical implementation of storage and retrieval processes.

Cognitive aspects of query languages were studied at the same time as cognitive aspects of programming, but the aim of the enquiry was different. The motivation was found in measuring the ease-of-use of query languages, from the point of view of end users (Reisner, 1981). One of the issues that were discussed in the early days is the procedurality of the query language. Some query languages specify more step-by-step methods to obtain results than others. SQL, today's standard, is a set-oriented language, and was been labeled "less procedural" or even non-procedural in the debate.

Cognitive Aspects of Software Development

Chan, Wei and Siau (Chan, Wei, & Siau, 1993) focus on cognitive processes of abstraction in the user-database interface. They distinguish three levels of abstraction: a conceptual level (a description of the user's world), a logical level (describing the database world) and a physical level (describing states in computer memory). Entity Relationship (ER) models describe the conceptual level (objects from the user's world). The relational model, relational algebra and SQL refer to the logical level. At the logical level, objects are held to exist in the user's mind. The physical level is not important to the user, because it is hidden by the RDBMS.

There are knowledge transformations between the conceptual level and the logical level, conceptual to logical transformations and logical to conceptual. The authors' conclusion is that to support understanding of how to write queries, the ER model (conceptual level) performs best.

Chan, Wei and Siau study naive users. De Haan and Koppelaars (de Haan & Koppelaars, 2007) explicitly address database professionals. Aside from writing queries, professionals need tools to control the RDBMS. RDBMSs conform to the relational model, which is why professional database engineers should master the logical level, and the database world's description in terms of set theory. Database professionals work with "objects" in the database world, a mathematical construct, based on set theory. No studies on the cognitive aspects of this kind of user-database interaction are known to us.

5.4 A NEED FOR EMPIRICAL STUDY

Computer programs can be characterized by their algorithm and by their abstract description (Pair, 1993). The question here is how programmers, designers and professionals characterize the abstract software structures they work with.

First of all, we observe that this characterization can change over time. Novice OO programmers mainly adopt a procedural strategy and start with a procedural representation of the topic they are considering (Detienne, Wiedenbeck et al.). Expert OO designers are able to switch between object view, emphasizing relations in the problem domain, and procedure view, emphasizing the execution of the program. According to Rist (as cited by Détienne, 1997), objects in OO design are an addition to the procedural structure of the system.

Procedurality of database languages has been the object of debates, but today database theory has a standard language, that can be characterized as less-procedural: SQL. Users of RDBMSs formulating queries need not know

Computing Education in a Hybrid World

how the software works. RDBMS software is a product that responds to requirements. Its accurateness in implementing the relational model is a measure of its quality. When a database professional detects behavior that is not conform with the model, he or she calls the vendor's helpdesk and reports a bug. The helpdesk sometimes offers an alternative strategy to avoid the problem (if available), unveiling parts of the RDBMS's structure. Expert database professionals know the software they work with better than their colleagues. Their queries perform better, and they are able to avoid deviant behavior of the RDBMS because they partially understand what is "inside the machine". Database experts, too, switch between abstract (relational) view and procedure view.

We are interested in mental models, instantiated by computing professionals while communicating with each other. These mental models are instantiated in WM, a structure having limited capacity. Even if experts of both disciplines seem to be able to switch between object view and procedural view of the software they handle, this does not mean that they can use the two views simultaneously, or that professionals are able to switch between views while communicating with colleagues.

We found different references to mental models in the literature concerning cognitive aspects of (OO) programming and user-database interaction. These are: (1) the notional machine, simulating an idealized computer, (2) the object view, where the individuation of (problem domain) objects guides the design activity, emphasizing static aspects of the solution, (3) the procedure-centred view, emphasizing the dynamics of a program and (4) a set-theoretical model, describing the problem domain in abstract terms and thus defining the database world. The notional machine and a procedure-centred view correspond to Schwank's functional thinking; set theory and the object-view correspond to predicative thinking.

We experience no contradiction between Schwank's view on algorithmic thinking and the findings we report in section 5.3 on cognitive aspects of OO design and user-database interaction. Terminology can be confusing, in particular where the term "functional" is used. Schwank (Schwank, Cognitive structures and cognitive strategies in algorithmic thinking, 1993) observes that in "functional thinking", the word "functional" does not refer to the mathematical term "function" intended as "a relation where each element of the domain (input) is related to exactly one element of the range (output)". Schwank classifies descriptions of algorithms in terms of input/output as "predicative thinking", thinking in terms of judgments. Détienné's "functional" strategy, where the definition of functions guides the design process, is not related to Schwank's "functional" thinking.

Cognitive Aspects of Software Development

Many efforts have been made to describe knowledge structures involved in programming. The distinction between a more descriptive dimension of knowledge and a dimension describing activities is omnipresent, just as the distinction between active and passive views on problems. The differences refer to structures in LTM; Schwank describes the relation between her theory and cognitive theories describing knowledge structures in LTM.

Triggered by Pennington's remark on the role of the language in the mental representation of expert FORTRAN and COBOL programmers, we ask ourselves why the FORTRAN programmers have become experts in FORTRAN instead of COBOL and vice versa. FORTRAN is known as a language that emphasizes the control flow of the program (Pennington, 1987), where COBOL was intended to be a business-friendly language, suitable to inexperienced programmers, and independent of knowledge of the underlying computer (Cobol, 2014). We experience no necessary contradiction between Pennington's results and Schwank's view on algorithmic thinking.

We found unsolved issues, in particular the oddity reported by D tienne. If OO design is driven by domain knowledge, why do novice designers experience difficulties in class creation? This could match Schwank's observation of functional thinkers: they are able to learn to accomplish "predicative" task successfully, but might need more time or perform worse than predicative thinkers.

We hypothesize differences between groups in the instantiated mental models used to handle a concept that is fundamental to both disciplines: the "object". We expect software engineers' mental models to match a functional approach and information managers' mental models to match a predicative approach.

We also expect to find different problem-solving preferences, directing the student toward one direction (a career in programming/software engineering) rather than another (a career in databases/information management).

5.5 A FIRST EXPERIMENT: HOW DO PROFESSIONALS UNDERSTAND THEIR SYSTEMS?

We designed a teach-back protocol to elicit information about the participants' mental models and recruited students of comparable age and level of education, enrolled in different computing curricula. The participants' answers are scored along categories, derived from the four

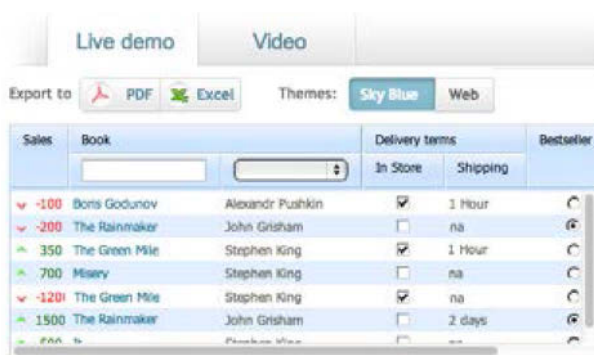
Computing Education in a Hybrid World

types of mental models mentioned in section 5.4. Three raters are involved in this experiment. Two are currently lecturing Computer Science in Dutch higher education, one is senior designer of documentation for scientific software. All the raters have a background in computer science and hold a Master's degree.

The protocol was tested in a pilot study with 5.6 participants, all lecturers of Computer Science in higher education. Based on the feedback, textual changes were made to the questions.

5.5.1 QUESTIONS

The protocol is introduced by a brief case description, an online bookstore (see Figure 5-1).



The screenshot shows a web interface for an online bookstore. At the top, there are tabs for 'Live demo' and 'Video'. Below these are 'Export to' options for PDF and Excel, and 'Themes' for 'Sky Blue' and 'Web'. The main content is a table with the following data:

Sales	Book		Delivery terms		Bestseller
			In Store	Shipping	
-100	Boris Godunov	Alexandr Pushkin	<input checked="" type="checkbox"/>	1 Hour	
-200	The Rainmaker	John Grisham	<input type="checkbox"/>	na	
350	The Green Mile	Stephen King	<input checked="" type="checkbox"/>	1 Hour	
700	Misery	Stephen King	<input type="checkbox"/>	na	
-120	The Green Mile	Stephen King	<input checked="" type="checkbox"/>	na	
1500	The Rainmaker	John Grisham	<input type="checkbox"/>	2 days	
2000	The Green Mile	Stephen King	<input checked="" type="checkbox"/>	na	

Figure 5-1 The context of the teach-back question: an online bookstore

The case description states that only information about books is relevant for our purposes, and that, therefore, the rest of the data is skipped. This results in a dataset with repeating rows having the same state. Participants are asked to count and describe the “Book-objects” they distinguish. No indications are provided for the choice of a theoretical context.

This situation is, in fact, ambiguous: the question can be answered by describing: (1) books in the bookstore’s assortment (objects in the problem domain), (2) Book-objects in permanent storage (in the programming domain) or (3) instances of Book-objects (in the programming domain)

Participants are instructed to explain in writing to an imaginary fellow student: (1) How many different Book-object they see and what these objects are, (2) What happens if the system is asked to produce additional information about one of the books. Up to now, we have been concentrating

Cognitive Aspects of Software Development

on the analysis of the first part of question (1): “How may different Book-objects do you see?”

5.5.2 SCORING CATEGORIES

To establish the number of different Book-objects, the rater scores one of the following answers:

- 4 objects (or 5, if the participant counts the last row, which is only partially visible),
- 6 objects (or 7, if the participant counts the last row),
- 0 (a number that is not traceable to Book-objects, or no number mentioned),
- B (both answers “4 objects” and “6 objects” are mentioned and explained).

5.5.3 PARTICIPANTS

The teach-back questions were answered by four groups of male Bachelor students, enrolled in professional curricula:

- 35 students attending 1st year classes Business IT & Management. Most of them enrolled in a professional curriculum in a computing discipline in 2013, 2 in 2010.
- 19 attending 3rd year classes Business IT & Management. Most enrolled in 2011, 1 in 2010, 1 in 2008.
- 18 attending 3rd year classes Computer Science, field of study Information Engineering. Most enrolled in 2011, 6 in 2010, 1 in 2009, 1 in 2002.
- 29 attending 3rd year classes Computer Science, field of study Software Engineering. Most enrolled in 2011. 4 in 2010.

All groups had attended at least one course “Relational Databases” and one course “Introduction to Programming in Java”. The Business IT & Management curriculum emphasizes modelling. The Computer Science curriculum concerns software development and has two fields of study. Software Engineering puts the emphasis is on programming, Information Engineering on the construction of business solutions (Hogeschool Utrecht, 2012).

5.5.4 HYPOTHESES

With this preliminary experiment, the following hypotheses are tested:

Computing Education in a Hybrid World

H1: “there is no significant difference between the conceptualization of the notion of ”object” reported by 1st and 3rd year students Business IT & Management”.

H2: “there is no significant difference between the conceptualization of the notion of ”object” reported by members of the following groups: 3rd year students Business IT & Management, 3rd year students Information Engineering and 3rd year students Software Engineering”.

5.5.5 PRELIMINARY RESULTS

The following categories were scored:

- 4: 47 participants. Many of them explain their choice (“There are 6 Book-objects, but two of them occur twice”), indicating that they are counting sets of objects.
- 6: 26 participants. Explanations vary from “n rows, n Book-objects” to “6 Book-objects. Some occur twice, but they are different objects” and “I am thinking OO-Java”.
- 0: 28 participants.

Category “B” was not scored in our samples. The answers are summarized in Table 5-1 and Table 5-2.

	4	6	0	n
1st year Business IT & Management	19	6	10	35
3rd year Business IT & Management	7	1	11	19

Table 5-1 number of Book-objects counted by students Business IT & Management

We found differences close to significance between the samples in Table 5-1 (chi sqr = 4.87; p<0.1) and reject H1. Most students of the 1st year Business IT & Management count 4 objects. They seem to interpret “objects” as problem domain entities or as database entries. 3rd year students seem to be less certain in their interpretation: their answer is scored more frequently “0”.

	4	6	0	n
3rd year Business IT & Management	7	1	11	19
3rd year Information Engineering	5	10	3	18
3rd year Software Engineering	16	9	4	29

Table 5-2 Number of Book objects counted by 3rd year students, sampled by curriculum

Cognitive Aspects of Software Development

We found significant differences between the samples in Table 5-2 (chi sq = 19.19; $p < 0.01$) and reject H2. We conclude that future Information Engineers seem to interpret “objects” as instances: the answer “6 objects” is predominant. Future Software Engineers show the opposite preference and count 4 objects. Business IT & Managements students seem to elude the question.

5.6 PRELIMINARY CONCLUSIONS

Our investigation shows at least two ways to characterize the abstract notion of “object” that are currently used by professionals. These characterizations are not compatible and lead to different judgements about “objects”. In the same situation, some professionals will identify 4 objects but others will perceive 6. The preference for 4 or 6 objects is not distributed ad random between professionals. We found indications for differences between groups of 3rd year students, enrolled in different computing curricula. Different preferences in the conceptualization of “object” can be a source of communication problems between groups of computing professionals. The lack of agreement about the definition of one of the basic notions of the discipline is alarming, just as the apparent difficulties to recognize this issue and to discuss it.

5.7 ACKNOWLEDGEMENTS

We thank the Hogeschool Utrecht and Johan Versendaal for their support; we thank the students and their lecturers for their cooperation.

6 Conceptualizations of the Notion of an Object⁴

Abstract— Computer Science has evolved towards a discipline with different branches. Software is designed, produced and linked taking into account different viewpoints. This process typically involves multidisciplinary teams: Front End Developers, (OO)Programmers, Database Engineers. Software developers, who were educated in different computing disciplines, meet on the shop floor, where they link together software that was designed from different viewpoints. In this paper, the emphasis is on human characteristics, rather than on the formal properties of programming and modeling languages. Do the involved computing practitioners refer to the same concepts when they use the same words? A preliminary version of this study (Benvenuti & van der Veer, PPIG2014, 2014) addressed the assessment of differences in mental representations of abstract entities involved in programming and modeling. In this extended version we report the results of an experiment, designed to compare mental representations of abstract concepts with mental models described in the literature. We point at differences between groups of students, enrolled in different computing curricula, and explore possible explanations.

KEYWORDS:

computer science education, engineering education research, human factors

6.1 INTRODUCTION

In the past couple of decades, Computer Science has developed different branches, each with its own body of knowledge and its own problem area. Different computing curricula have their *raison d'être* in different approaches to the digitalization of information. What happens when the paradigms meet, when engineers link together software that was designed from different points of view? This area is characterized by persistent problems, such as the Object/Relational impedance mismatch.

⁴This work was originally published as: Benvenuti, L., Louwe Kooijmans, C.F., Versendaal, J., & van der Veer, G.C. (2015). Representations of abstract concepts, differences across computing disciplines. *Frontiers in Education 2015*. El Paso, TX. ©2015 IEEE, ISBN :978-1-4799-8454-1 ,doi: 10.1109/FIE.2015.7344411

Computing Education in a Hybrid World

Many attempts have been made to solve these problems by developing new software, such as Object Relational Mappings. This is problematic, though. The problem with these solutions is their validation. Multidisciplinary teams that work on tangible artifacts can discuss advantages and disadvantages of machine control software, plant engineering software or computer graphics software, knowing that they refer to the same tangible or visible artifacts. But many computing professionals work exclusively on abstract entities. Technical solutions of the Object Relational Impedance Mismatch can only be evaluated if practitioners perceive a common field of application, if they work with the same abstract entities.

The new ACM/IEEE guidelines for the undergraduate program in Software Engineering (SE) (ACM/IEEE, 2014) acknowledge the problem of working with abstract entities and the challenges it involves for “knowledge exchange during the process of [software] design” (pg. 11). Although we acclaim the attention for this phenomenon, we regret the location of the problem inside the SE discipline. We suspect practitioners from other branches than SE to conceptualize abstract entities differently than SE. More generally, we suspect different computing disciplines to operate from different conceptualizations.

We will focus on the cognitive backgrounds of the issue, rather than its pedagogical perspective, although we will touch on educational implications. This paper describes an experiment to assess differences in mental representations of abstract entities, involved in programming and modeling. We recruited students of comparable age and level of education, enrolled in different computing curricula, and asked them to help us understand the way they conceptualize abstract concepts. In a preliminary (work-in-progress) version of this study (Benvenuti & van der Veer, PPIG2014, 2014), we reported incompatible ways to characterize the abstract notion of “object”. We also found indications for group preferences: significant discrepancies between participants, enrolled in different curricula.

This extended paper focuses on the explanation of the differences we have identified. We compare the participants’ mental representations with mental models described in the literature. We provide an overview of the research on cognitive aspects of programming and database interaction and explore the notion of individual preferences in mathematical problem solving (Schwank, Cognitive structures and cognitive strategies in algorithmic thinking, 1993).

Conceptualizations of the Notion of an Object

In section 6.2 we explore the notion of a mental model. In 6.3 we give an overview of research on cognitive aspects of programming and database interaction, followed by discussion in section 6.4. In section 6.5 we design an experiment to assess possible differences in mental models between senior students, enrolled in different computing curricula. Results are given in section 6.6, followed by a reflection in section 6.7, conclusions in section 6.8 and relevance in section 6.9.

6.2 BACKGROUNDS

6.2.1 MENTAL MODELS

According to Craik, the mind constructs “small-scale models” of reality and uses them to anticipate events (Johnson-Laird, *Mental Models*, 1989) (van der Veer & Puerta Melguizo, *Mental Models*, 2002) Johnson-Laird formulated a theory of mental models, meant to explain human thinking and reasoning. People, states Johnson-Laird, do not only apply inference rules while reasoning; they also consider the semantic content of the problems they are solving. Norman (Norman, 1983) is more specifically concerned with mental models in Human-Computer Interaction. Users, states Norman, construct mental models of computer systems incrementally, while interacting with systems. The resulting models are constrained by the users’ prior knowledge, needs and context. These models are often incomplete. They are limited by the human information processing system, by experience and by needs that can be contrasting: the need to focus and the need to retain important details. They are parsimonious in order to reduce mental complexity. User mental models are unscientific and unstable. They evolve over time: people learn, people forget. People use metaphors to simplify their models. Nevertheless, mental models are functional to support tasks such as planning, execution, assessment of results and understanding of unexpected events.

There is little agreement on the exact definition of the term “mental model”. Does the term refer to temporary structures in Working Memory (WM) or knowledge structures in Long Term Memory (LTM)? Cañas and Antolí (Cañas, 1998) introduce this definition: a mental model is “*the dynamic representation that is formed in WM combining the information stored in LTM and the extracted information from the environment*”.

Knowledge, in particular knowledge concerning computing paradigms, is acquired by education and experience and stored in LTM as knowledge chunks that evolve in time. According to Norman, evolution is not

Computing Education in a Hybrid World

necessarily positive: improvement is possible, deterioration also. These knowledge chunks can only be used if they are retrieved and instantiated in WM, in what we will call a mental model from now on, following Cañas' and Antolf's definition.

Human WM is limited. People take these limitations into account and adopt strategies to keep mental models manageable in WM. The question here is: can we assume mental models of practitioners with different backgrounds to be compatible with each other?

6.2.2 INDIVIDUAL PREFERENCES

Schwank (Schwank, Cognitive structures and cognitive strategies in algorithmic thinking, 1993) studies the way people select mental models to represent mathematical knowledge. Schwank distinguishes a *predicative* and a *functional* approach to mathematic thinking. Predicative thinking is "static grasping", thinking in terms of judgments and relations; functional thinking is "dynamic grasping", thinking in terms of actions, processes and their effects. Schwank adopts this metaphor: predicative thinking is like solving puzzles, where one looks for (static) patterns. Functional thinking is mechanical thinking, like using gear: the focus is on what happens next. Most mathematical problems can be solved either way; this also applies to many problems in computer science.

Schwank used Raven's progressive matrices to assess problem solving preferences (Raven, cited by Schwank). 3x3 matrices lacking the lower right symbol were presented to the participants, which were asked to *construct* the last symbol to complete the sequence (instead of *choosing* the last symbol as is requested in Raven's intelligence test). The participants were asked to argue why they draw that particular symbol. Schwank found support for the thesis that the preference for a predicative or functional mode in problem solving is a stable individual characteristic.

6.2.3 ASSESSING MENTAL MODELS: THE TEACH-BACK PROTOCOL

Mental models cannot be observed directly. Many authors draw conclusions about respondent's mental models based on their behavior (Moray, 1998). Van der Veer adopts a different strategy: he asks the participants. Van der Veer (van der Veer, Learning, individual differences and design recommendations, 1990) extends an hermeneutic method designed by G. Pask, intended to elicit information about mental models (the Teach-Back method), and adopts it to detect differences in mental representations of users interacting with computers.

Conceptualizations of the Notion of an Object

A situation is simulated where the respondent has to interact with a computer. He/she is asked to explain the computer's functioning to an imaginary counterpart, a colleague or a student, who has similar experience with the situation. The questions are designed to activate both declarative and procedural knowledge structures, to obtain an overall picture of the participants' mental models. Questions are presented on white sheets of paper, and the participants are instructed to express themselves in whatever way they consider most adequate: text, drawings, keywords, diagrams etc. In this manner, the participants are encouraged to externalize the mental model they made of the situation. Next, the protocols are scored "blind", along pre-defined scoring categories in order to map the respondent's mental representations. Rating implies (1) reading the protocol in its entirety and trying to understand fully what it says; (2) trying to formulate how the participant understands the space of the teach-back question and (3) classify the responses into relevant categories for the purpose of the study. Rating the answers is a complex task: the rater has to interpret the participant's intention and classify it by means of scoring rules. This task requires considerable training. In order to safeguard reliability, two or more persons score the answers independently.

6.3 LITERATURE REVIEW

6.3.1 OBJECTS

One of the basic notions of software development is the "object". According to Booch, Rumbaugh and Jacobson (Booch, Rumbaugh, & Jacobson, 2005) an object is "A concrete manifestation of an abstraction; an entity with a well-defined boundary and identity that encapsulates state and behavior; an instance of a class". Objects are software constructs that can refer to real-world entities, entities that are also called objects (although some authors use the term domain objects (Larman, 2005)). A class describes a set of objects (Booch, Rumbaugh, & Jacobson, 2005). Equivalent objects can coexist during program running: instantiation generates bags of objects. Nevertheless, the term instance is seen as synonym with object (Booch, Rumbaugh, & Jacobson, 2005)

Handling these different natures of objects requires switching between mental models. In the next sections, we will review the literature on understanding of programming constructs.

Computing Education in a Hybrid World

6.3.2 COGNITIVE ASPECTS OF (OO) PROGRAMMING

Robins, Rountree and Rountree (Robins, Rountree, & Rountree, 2003) provide us with an extensive literature review of research concerning learning and teaching programming between 1970-2003. In a survey study, Détienne (Detienne, 1997) reviews empirical research on OO design and assesses claims about the cognitive benefits of the OO paradigm.

Robins, Rountree and Rountree mention different kinds of mental models involved in learning programming: mental models of the “task, problem or specification” that has to be solved by the program, mental model of the programming language, and mental models of the behavior of the running program. Many studies have noted a central role played by a model (or abstraction) of the computer. Du Boulay et al. (du Boulay, O’Shea, & Monk, 1989) call this the “notional machine”, an idealized, conceptual computer that is defined with respect to the language. Novices should develop an appropriate notional machine to master a programming language: the notional machine underlying Pascal is very different from the machine underlying PROLOG. A study by Mayer (Mayer, 1989) confirms that students supplied with a notional machine model perform better while solving some problems than students who are not given the model.

A model of program comprehension was provided by Pennington (Pennington, 1987). Comprehension occurs in the context of a problem domain. The program’s text is re-organized in mental representations with help of available knowledge structures. Pennington studies expert FORTRAN and COBOL programmers, and finds significant differences in their performances of comprehension tasks.

The OO approach has claimed to make modeling the problem domain easier for programmers. Détienne investigated this claim. She found that novice programmers have difficulties in class creation. According to Détienne, novices need to start with a procedural representation of the situation. This seems to indicate that knowledge is organized in terms of procedures, not in terms of objects and relations. But for expert OO designers, the claims find support. Experts do analyze problems in terms of objects and their relationships. Also the claim that OO designers design solutions that are closer to the problem domain is supported. Expert OO designers seem to shift between object view and procedure view.

Détienne comments on the oddity of the situation: if OO design is driven by domain knowledge, then the biggest benefit should be observed in novices; but this is not the case.

6.3.3 COGNITIVE ASPECTS OF USER-DATABASE INTERACTION

Today, the Relational Model is the leading model in database theory. There is no need to know how data are stored in a Relational Database in order to write a query: knowledge of the (abstract) data model of the database is sufficient.

Cognitive aspects of query languages were studied at the same time as cognitive aspects of programming (Reisner, 1981). One of the issues was the procedurality of the query language. Some query languages specify more step-by-step methods to obtain results than others. SQL, today's standard, is a set-oriented language, and was been labeled "less procedural" or even non-procedural in the debate.

Chan, Wei and Siau (Chan, Wei, & Siau, 1993) focused on cognitive processes of abstraction in the user-database interface. They distinguished three levels of abstraction: a conceptual level (a description of the user's world), a logical level (describing the database world in in set-theoretical terms) and a physical level (describing states in computer memory). The authors concluded that naïve users' understanding of how to write queries was best supported by a model of the conceptual level.

De Haan and Koppelaars (de Haan & Koppelaars, 2007) explicitly address database professionals. Professionals need to control the RDBMS, which is why professional database engineers should master the logical level, and the database world's description in terms of set theory. Database professionals work with "objects" in the database world, a mathematical construct, based on set theory. No studies on the cognitive aspects of this kind of user-database interaction are known to us.

6.4 A NEED FOR EMPIRICAL STUDY

The question here is how programmers, designers and professionals characterize the abstract software structures they work with. We observe that this characterization can change over time. Novice OO programmers mainly adopt a procedural strategy and start with a procedural representation of the topic they are considering (Detienne, 1997) Expert OO designers are able to switch between object view, emphasizing relations in the problem domain, and procedure view, emphasizing the execution of the program.

We are interested in mental models, instantiated in WM by computing professionals while communicating with each other. WM has limited capacity. Experts seem to be able to switch between object view and procedure view. This does however not mean that they can use the two

Computing Education in a Hybrid World

views simultaneously, or that they are able to switch while communicating with colleagues.

We found multiple references to mental models in the literature concerning cognitive aspects of (OO) programming and user-database interaction. These are: (1) the notional machine, simulating an idealized computer, (2) the object view, where the individuation of (problem domain) objects guides the design activity, emphasizing static aspects of the solution, (3) the procedure-centered view, emphasizing the dynamics of a program and (4) a set-theoretical model, describing the problem domain in abstract terms and thus defining the database world. The notional machine and a procedure-centred view correspond to Schwank's functional thinking; set theory and the object-view correspond to predicative thinking.

We found references to discrepancies between professional groups. Pennington reports of expert COBOL and FORTRAN programmers performing differently on program comprehension tasks.

We also found unsolved issues, in particular the oddity reported by Détienne. If OO design is driven by domain knowledge, why do novice designers experience difficulties in class creation? This could match Schwank's observation of functional thinkers: they are able to learn to accomplish "predicative" task successfully, but might need more time or perform worse than predicative thinkers.

We hypothesize differences between professional groups in the instantiated mental models used to handle a concept that is fundamental to most computing disciplines: the "object"

6.5 EXPERIMENT DESIGN

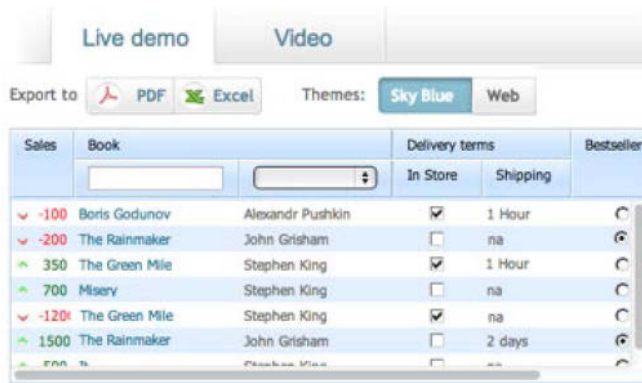
Do future computer professionals perceive sets of objects or bags of objects (instances)? Do they locate "objects" in the problem domain or in the programming domain? Do they think of software in terms of processes in the notional machine (sequences of events, communication between objects) or in terms of structures (attribute values, relationships, queries)?

We designed a Teach-Back protocol to elicit this information, and enquired students enrolled in different computing curricula. The participants' answers were scored along categories, derived from the four types of mental models mentioned in section 6.4. Four raters were involved in this experiment. Three have been lecturing Computer Science in Dutch higher education for over 20 years; one is senior designer of documentation for scientific software. All the raters have a background in Computer Science and hold a Master's degree or equivalent.

Conceptualizations of the Notion of an Object

6.5.1 CONTEXT

The protocol is introduced by a brief case description, an online bookstore (see Figure 6-1). It states that only information about books is relevant for our purposes, and that, therefore, the rest of the data is skipped. This results in a dataset with repeating rows having the same state. Participants are asked to describe the “Book-objects” they distinguish. No indications are provided for the choice of a theoretical context.



The screenshot shows a web interface for an online bookstore. At the top, there are buttons for 'Live demo' and 'Video'. Below these are 'Export to' options for PDF and Excel, and 'Themes' for 'Sky Blue' and 'Web'. The main content is a table with the following columns: Sales, Book, Delivery terms, and Bestseller. The 'Book' column is further divided into two sub-columns. The 'Delivery terms' column has sub-columns for 'In Store' and 'Shipping'. The table contains several rows of book data, including titles like 'Boris Godunov', 'The Rainmaker', 'The Green Mile', 'Misery', and 'The Green Mile' again, with authors like Alexandr Pushkin, John Grisham, and Stephen King. Each row has a 'Sales' value, a 'Bestseller' status, and a 'Delivery terms' section with checkboxes for 'In Store' and 'Shipping'.

Sales	Book		Delivery terms		Bestseller
			In Store	Shipping	
-100	Boris Godunov	Alexandr Pushkin	<input checked="" type="checkbox"/>	1 Hour	
-200	The Rainmaker	John Grisham	<input type="checkbox"/>	na	
350	The Green Mile	Stephen King	<input checked="" type="checkbox"/>	1 Hour	
700	Misery	Stephen King	<input type="checkbox"/>	na	
-120	The Green Mile	Stephen King	<input checked="" type="checkbox"/>	na	
1500	The Rainmaker	John Grisham	<input type="checkbox"/>	2 days	

Figure 6-1 Online bookstore

The situation is, in fact, ambiguous: the question can be answered by describing: (1) Book-objects in the problem domain (books in the bookstore’s assortment), (2) Book-objects in the programming domain (Book-objects in permanent storage), (3) Book-objects in the programming domain (instances of Book-objects), (4) objects in the programming domain (elements of the HTML-document).

6.5.2 QUESTIONNAIRE

We asked the participants, how many different Book-objects they saw in the table in Figure 6-1. To answer the question, a mental model had to be instantiated.

Participants were presented with white sheets of paper and were instructed to explain in writing to an imaginary fellow student:

How many different Book-objects they saw and what these objects were,
What would happen if the system was asked to produce additional information about one of the books.

Computing Education in a Hybrid World

These questions activate declarative and procedural knowledge structures (section 6.2.3). A last question aims to obtain an indication of the respondent's preference for predicative or functional thinking. Following Schwank's method, we used a matrix from Raven's test (Figure 6-2) as explained in section 6.2.2. We asked participants:

to add the missing symbol and to explain how they had constructed the solution.

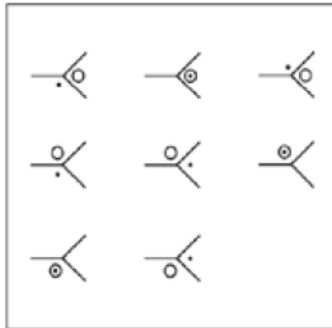


Figure 6-2 The matrix used to math individual problem solving preferences

6.5.3 RESEARCH QUESTIONS AND SCORING CATEGORIES

This experiment was designed to answer the following research questions.

Are there differences across the disciplines in the amounts of objects counted by participants?

The aim of research question a) is to identify possible communication problems between computing professionals. Scoring was based on the participants' answers to the first part of question 1 of the questionnaire (how many different objects do you see?).

Although differences in amounts of objects can indicate differences in mental models, their occurrence is not sufficient to draw conclusions. Research question b) focuses on the participants' mental model.

a. Are there differences across the disciplines in the mental models, instantiated by participants?

Scores concerning questions b were based on the interpretation of the participant's answers to both questions 1 and 2 of the questionnaire. To determine the mathematical structure envisioned by participants (Set or Bag, see sections 6.3.1 and 6.3.3) and the participants' approach to software

Conceptualizations of the Notion of an Object

(object view or procedure view, see section 6.3.2), we defined the following sub-questions:

b1. Does the participant envision a set of Book-objects or a bag of Book-objects (instances)?

b2. Does the participant make assumptions about the software implementation?

b3. If the participant describes the programming domain, on which aspects does s/he focus?

Different problem solving preferences might reflect on the approach to software, resulting in different mental models. We investigated the participants' problem solving preference, as defined by Schwank (see section 6.2.2).

c. Which problem solving strategy adopts the participant to construct the solution of question 3?

For each research question and sub-question, scoring categories were defined.

a) To analyze discrepancies between the numbers of Book-objects counted by the participants, we scored:

- 4 objects (or 5, if the participant counts the last row, which is only partially visible),
- 6 objects (or 7, if the participant counts the last row),
- A: a number that is not traceable to Book-objects,
- N: no number mentioned,
- Both: both answers "4 objects" and "6 objects" are mentioned and explained

b) To determine the participants' mental models, we analyzed the answers to research questions b1..b3.

b1. Does the participant envision a Set of Book-objects or a Bag of Book-objects?:

- S: a set of objects
- B: a bag of objects (instances)
- N: none of the above answers are supported by the participant's description
- BS: participant sees both possibilities, a Bag of objects and a Set of objects

Computing Education in a Hybrid World

Rating the dominant aspect of software description (b3) is only possible if the participant describes the software. Many participants did not. We rated the granularity of software description (b2), and – if the software was described in some detail – the dominant aspect.

b2. To determine the granularity of software description, we scored:

- Macro: the participant describes the system in terms of its macro-structure, or as a black box. No details about the software's implementation are added.
- Detail: the participant adds detailed hypotheses about (part of) the software's implementation.

b3. To determine the dominant aspect in the description of software, we adopted the following categories:

- S: Participant focuses on static aspects of software (Détienne's object view)
- P: Processes are also described (Détienne's procedure view)
- M: Unclear. Participant provides mixed information

c) To determine the participant's problem solving strategy, we used Schwank's categories:

- P: Predicative thinking. Participant refers to the location of symbols to justify the solution he proposes
- F: Functional thinking. Participant perceives symbols as entities that can move, and performs motion analysis to justify the solution he proposes
- N: No choice could be made

6.5.4 RELIABILITY

Scoring was done by two raters at the time. The raters interpreted the participant's answers and scored them along the categories we discussed in section 6.5.3. The raters had been trained by scoring 6-10 questionnaires, followed by more sessions where interpretation issues were discussed. Still, discrepancies between raters do occur where interpretation is involved. The degree of agreement (inter-rater reliability) is expressed by Cohen's Kappa coefficient (Bryman, 2012). Generally, a coefficient above 0.75 is considered very good; between 0.6 and 0.75 as good and between 0.4 and 0.6 as fair (page 280). Reliability was calculated on the questionnaires that had not been used for training purposes.

Conceptualizations of the Notion of an Object

After scoring the categories and measuring the reliability, the discrepancies were discussed between the raters. If agreement was possible, the discrepancies were solved. In the other case, the answer was scored as “N” (research questions a, b1 and c) or “M” (research question b3). All issues concerning research question b2 could be solved.

6.5.5 PARTICIPANTS

The teach-back questions were answered by five groups of male Bachelor students, enrolled in professional curricula:

- 35 attending 1st year classes of Business IT and Management. Most of them enrolled in a professional curriculum in a computing discipline in 2013, 2 in 2010. They were not included in our samples.
- 19 attending 3rd year Business IT and Management, enrolled in in a professional curriculum in a computing discipline in 2011. The exceptions: 1 enrolled in 2010, 1 in 2008.
- 18 attending 3rd year of Computer Science, field of study Information Engineering. Most enrolled in 2011. The exception: 6 enrolled in 2010, 1 in 2009, 1 in 2002.
- 29 attending 3rd year of Computer Science, field of study Software Engineering. Most enrolled in 2011. 4 enrolled in 2010.
- 4 enrolled in other computing curricula. They were not included in our sample.

All groups had attended at least one course “Relational Databases” and one course “Introduction to Programming in Java”. The Business IT and Management curriculum emphasizes modelling. The Computer Science curriculum concerns software development. Computer Science has two fields of study: Software Engineering puts the emphasis is on programming, Information Engineering on the construction of business solutions (Hogeschool Utrecht, 2012) (Hogeschool Utrecht, 2013)

6.5.6 HYPOTHESES

Different computing disciplines seem to allow, or even to promote, fundamentally different conceptualizations of the “object”. These differences between disciplines can be systematic. Systematic differences would lead both to different conceptualizations and to different amount of objects detected, therefore to systematically divergent frequencies of the answers

Computing Education in a Hybrid World

to the questions in section 6.5.2. We cannot formulate expectancies about the magnitude of this effect.

To test the effect, we will investigate if we can reject the following null hypotheses in favour of plausible alternative hypotheses. We will reject the null-hypotheses if the probability p of finding our results less is than 0,05. In that case, we will conclude that the deviation from the null hypothesis is significant. We will state that we have found significant differences if the deviation is conform our expectation.

We will test the following null hypotheses:

Ha: *“there is no difference between the amount of different “objects” reported by members of the following groups: 3rd year students Business & IT Management, 3rd year students Information Engineering and 3rd year students Software Engineering”.*

Hb: *“there is no difference between the conceptualization of the notion of “object” reported by the members of the following groups: 3rd year students Business & IT Management, 3rd year students Information Engineering and 3rd year students Software Engineering”*

Hypothesis Hb can be divided in

Hb1: *“there is no difference between the structures (set or bag) envisioned by the members of the groups mentioned above.”*

Hb2: *“there is no difference in granularity of description of the software between the members of the groups mentioned above.”*

Hb3: *“there is no difference in the aspects, emerging from the software descriptions of members of the groups mentioned above.”*

Hc: *“there is no difference between the problem solving preferences of the members of the following groups: 3rd year students Business & IT Management, 3rd year students Information Engineering and 3rd year students Software Engineering”.*

6.6 RESULTS

6.6.1 HA: AMOUNTS OF DIFFERENT OBJECTS REPORTED

From the answers to question 1 of the questionnaire: “How many different Book-objects does the participant count?”, the following categories were scored:

- 4: 27 participants. Many of them explain their choice (“There are 6 Book-objects, but two of them occur twice, hence 4 different Book objects”, “there are 4 Book objects, since some are double”).

Conceptualizations of the Notion of an Object

- 6: 19 participants.. Explanations vary from “n rows, n Book-objects” to “6 Book-objects. Some occur twice, but they are still different objects” and technological explanations (“I am thinking OO-Java”).
- A: 8 participants mention an amount of objects that is not traceable to Book-objects: they count 1,2,3,4,12,14 or 16 objects. They describe: classes, attributes, User Interface objects (e.g. search-fields) or attribute values.
- N: 12 participants do not answer the question. 4 out of these 12 give elaborate answers, incidentally illustrated with diagrams, without counting the objects (e.g. “7 books, described with title and author. 2 of them occur twice”).
- B: 0 participants. The answer: “We see 6 instances of 4 Book-objects” is never given. One student counts “6 instances of 4 Books”. He is one of the participants that were not included in our sample (see section 6.5.5).

	Number of objects			n
	4	6	0	
3 rd BIM	7	2	10	19
3 rd IE	5	10	3	18
3 rd SE	15	7	7	29

Table 6-1 Number of Book-objects, sampled by curriculum

Discrepancies between raters were very limited (Cohen’s Kappa = 0,795). Agreement on 1 answer was not possible; it was scored “N”, as indicated in section 6.5.4.

The scores of question a) are shown in Table 6-1. Column “O” (“Other”) summarizes the categories “A” (a number, not referring to Book-objects) and N (not a number). Category “B” was not scored in our samples.

We found significant differences between the samples in Table 6-1. ($\chi^2 = 13,1$; $p < 0,05$) and reject H_0 . Students Software Engineering seem to count more often 4 objects. Students Information Engineering seem to prefer counting 6 objects. Student Business IT and Managements seem to elude the question.

Computing Education in a Hybrid World

6.6.2 HB: MENTAL MODELS ACROSS THE DISCIPLINES

To determine differences in mental model, the participants' answers to the questions about the system (questions 1 and 2 of the questionnaire) were read in totality and interpreted. This could only be done if the participant had answered at least one of these. This was not the case for two participants; one of them was included in our sample. Questions b1..b3 were scored for 17 students, enrolled in the Information Engineering curriculum, instead of 18.

The following categories were scored for research question b1 ("Does the participant envision a set of objects or a bag of objects?"):

- Set: 26 participants. ("There are 4 Book objects, because some of them are double", "6 records representing 4 Book objects").
- Bag: 25 participants ("6 instances", "12 different objects, Books and Authors")
- N: 14 participants. this category was scored if question 1 was not answered, if no useful answer was provided (e.g. "2 Book-objects") or if the answer was contradictory ("Book objects are identified by their attributes: ID, title, author. I see 4 Book-objects with different ID's")
- BS: this category was not scored.

Some discrepancies between raters occurred (Cohen's Kappa = 0.597). Agreement was not possible on 2 answers in this sample. They were scored 'N'.

The meaning of the term "different" is ambiguous in OO-programming. Equivalent instances can refer to the same object. We found 3 answers reflecting this dilemma: "6 Book-objects, but two of them occur twice, hence 4 different Book objects". They were scored as "Bag".

	Set or Bag			n
	<i>Set</i>	<i>Bag</i>	<i>N</i>	
3 rd BIM	5	7	7	19
3 rd IE	6	11	0	17
3 rd SE	15	7	7	29

Table 6-2 Set or Bag, sampled by curriculum

The answers to research question b1 are shown in Table 6-2. We found significant differences between the samples in Table 6-2. (Fisher's exact test,

Conceptualizations of the Notion of an Object

$p = 0,0101279$) (Soper, sd) and reject H_{b1} . Students Software Engineering seem to show a preference for the structure “Set”, where students Information Engineering appear to work with “Bag”s of objects. Students Business IT and Management are random divided between the options.

The case provides minor structural information about Book objects, in Figure 6-1. While scoring b2 and b3, the raters focused on information that was added by the participants, information that could not be traced back to the case or the questions.

Discrepancies between raters occurred (Cohen’s Kappa = 0.576). Agreement was not possible on the scores on question b3 for the answers of 5 participants. They were scored “M”.

The following categories were scored for research question b2 (assumptions about the software implementation):

- Macro: 18 participants roughly described the software they had envisioned without adding structural or process information. (“the name of the book is looked up in the database”, “If the user points at the row, the system displays information”).
- Detail: 48 participants provided more detailed information (list of attributes, class diagram, code, activity diagram, query)

	Granularity of software description		
	Macro	Detail	n
3 rd BIM	9	10	19
3 rd IE	5	12	17
3 rd SE	3	26	29

Table 6-3 Granularity of software description, sampled by curriculum

The scores of question b2 are summarized in Table 6-3. Approximately 90% of the Software Engineering students provided implementation information, against ca. 70% of the Information Engineering group and 50% of the Business IT and Management group. The differences are significant ($\chi^2 = 8,21$, $df=2$, $p < 0,05$). We reject H_{b2} .

The following categories were scored for research question b3 (“dominant aspects in the description of software”):

- S: 28 participants mainly added structural information (attributes or relationships, class diagram, query)

Computing Education in a Hybrid World

- P: 14 participants mainly added information about procedures (activity diagram, sequences of events, communication between software components)
- M: 6 participants added both kinds of information

The scores of question b3 are summarized in Table 6-4. We found no significant differences between the samples in Table 6-4. (Fisher's exact test, $p=0,12682945$) (Soper, sd)

	Dominant aspects in software description			n
	<i>S</i>	<i>P</i>	<i>M</i>	
3 rd BIM	9	1	0	10
3 rd IE	5	6	1	12
3 rd SE	14	7	5	26

Table 6-4 Dominant aspects in software description, sampled by curriculum

We found significant differences in mental models of students enrolled in different curricula, and reject H_0 . Apparently, there are differences in the way future professionals conceptualize software.

Some of the participants appear to be in doubt (it depends on the interpretation of the figure, on the relationship between title and author, etc.), but seem not to be able to give expression to their doubts.

6.6.3 HC: PROBLEM SOLVING PREFERENCES

For research question c, the following categories were scored:

- P: Predicative. ("the circle's position is the same in every row, the dot's position is the same in every column", or: "I choose the missing symbol")
- F: Functional ("In each row, the dot moves counterclockwise while the circle stays at the same place")
- N: No choice could be made

Discrepancies between raters were limited (Cohen's Kappa = 0.742). Agreement could not be reached on 2 answers; they were scored "N".

The scores of question c are summarized in Table 6-5. We found no significant differences between the samples in Table 6-5. (Fisher's exact test, $p=0,25695717$) (Soper, sd)

Conceptualizations of the Notion of an Object

	Problem solving preference			n
	<i>P</i>	<i>F</i>	<i>N</i>	
3 rd BIM	6	7	6	19
3 rd IE	5	11	2	18
3 rd SE	14	11	4	29

Table 6-5 Problem solving preferences, sampled by curriculum

6.7 LESSONS LEARNED

Our initial ambition was to classify mental representations of “objects”, in problem domain objects, stored objects or instances. We partially succeeded. We were able to underpin some observations about the participants’ mental models. We determined the mathematical settings participants refer to while working with objects (set or bag), and assessed in which terms they envision software while communicating with colleagues (macro or detail). No conclusions could be drawn about the location of the objects in problem domain or programming domain. This might be due to different causes. One applies to the casus. Students know bookstore-cases from their courses on Databases. The choice for a bookstore might have appealed to previous knowledge, pointing towards a possible interpretation for the term “object”. Also, the formulation of question 2 of the questionnaire (“What happens if the system is asked to produce additional information about one of the books?”) might have suggested a macro approach to the system, reinforcing that interpretation. We will take this experience into account for the new version of the questionnaire.

Interpreting student mental models has proven to be challenging, even for experienced assessors of students’ work. The raters are puzzled by the measure in which their interpretations can differ. One example is: although “set” and “bag” are well-defined mathematical concepts, some of the participants’ answers gave cause for discussion. One student explained how he counted: “Objects are different versions. E.g. [if we have]: 1,2,1. 1 and 1 are the same: 2 different objects”. The raters (one has an engineering background, one is a former database professional) scored respectively “Bag” (different objects: 1 and 1), and “Set” (different objects: 1 and 2). Both understood the other’s point of view, but agreement could not be reached. Rating in teams of mixed expertise appears to be crucial in this experiment. Structured interviews with participants may be necessary in the future to fully understand some of the answers.

6.8 CONCLUSIONS

Our investigation showed at least two ways to characterize the abstract notion of “object” that are currently used by professionals. Both characterizations are correct, but they are not compatible: they lead to different judgments on the number of “objects”. Some participants identified 4 Book-objects, others identified 6 Book-objects. The preference for 4 or 6 objects was not distributed at random. Apparently, Software Engineers are more likely to see 4 objects, in contrast with most Information Engineers, who show a preference for detecting 6 objects.

One third of the participants could not provide information about the amount of objects. Although some of them expressed doubts about the question, only one of them was able to point in the direction of possible ambiguity of the notion of “object”.

We found indications for the instantiation of different mental models across the disciplines. SE, IE and BIM students in our study seem to differ in their perception of the appropriate structure for objects (set or bag). More in general, they appear to differ in the measure in which they envision a possible implementation while discussing software. Further research is needed to better describe computing professionals’ mental models, and investigate differences across the disciplines.

We did not find significant differences in the way different groups approach software (object view or procedure view). No significant differences between individual problem solving preferences were found across the groups either. This might indicate that the differences we describe are acquired, hence a responsibility of designers of computing curricula.

6.9 OUR MESSAGE FOR EDUCATION

The immediate relevance of our work is in curriculum design. We recommend all computing curricula to explicitly cultivate awareness for different approaches to computing concepts. Based on this study, we also recommend educators and practitioners to establish and use a refined terminology for the notion of “object”. A refined terminology will improve recognition of different mental models. Whether it will support computing professionals sufficiently in handling different mental models, is an issue that needs to be researched.

The lack of agreement about the definition of one of the basic notions of the computing discipline, we have found between students enrolled in

Conceptualizations of the Notion of an Object

different curricula, is worrisome, just as the apparent difficulties to recognize this issue and to discuss it properly.

ACKNOWLEDGMENT

We thank raters Jikke van Wijnen and Brigit van Loggem. We thank the students and lecturers of the HU University of Applied Sciences Utrecht for their cooperation.

PART II – Conclusions

In this part, we have investigated if students, enrolled in different undergraduate curricula, conceptualize abstract entities in the same way. We had hypothesized differences in mental models for the notion of object between students Software Engineering (functional approach) and students Business IT and Management (predicative approach).

RESEARCH QUESTION RQ4

RQ4: Do students, who were educated in different computing disciplines, develop the same mental models for the abstract concepts they work with? I.e., are different approaches to computing interchangeable?

We inquired differences across computing disciplines between conceptualizations of software, in particular: (1) differences in numbers of “objects” counted, (2) problem solving preferences of students and (3) conceptualizations of the notion of “object”. We found indications for differences between disciplines, but the differences we found were not the differences we had hypothesized.

The samples we were able to collect were small, interrater reliability of the scores varied between 0.795 (very good) and 0.576 (fair). We will discuss our findings in descending order of reliability;

(1) Within the limits set by our sample size, we found reliable indications for differences between the way students, enrolled in different computing curricula, conceptualize the abstract notion of “object”. The numbers of objects students counted appear to be different across the disciplines in our samples.

(2) We did not find significant differences in problem solving preferences (functional/predicative) across the groups.

(3) Although less reliably than the indications listed above, we found indications for differences between the structures the students refer to while counting objects (sets or bags). We found indications of differences in the measure in which students, enrolled in different computing curricula, envision possible implementations of the “object” while discussing software. We did not find significant differences in the way students approach software (object view or procedure view).

We had started our investigation by hypothesizing differences between groups of students in the instantiation of mental models for the “object”. We had expected to find differences in problem solving preferences across these

Computing Education in a Hybrid World

groups (section 5.4), and had expected Software Engineers' mental models to match a functional approach and Information Managers' mental models to match a predicative approach. This did not occur. We will return to these unexpected results in section 10.3.1.

Overall, we did not find awareness about the possible ambiguity of the abstract notion of "object". In our opinion, this is an issue education should address.

RESEARCH QUESTIONS RQ6, RQ7

RQ6 Which subject-specific strategies were recommended in the past?

One of the fundamental constructs in present-day's software development is the "object". Objects are often introduced informally, as software constructs that can refer to real-world entities. Depending on the context, the term can refer to real-world entities, or to software constructs. When it refers to software constructs, it can indicate persistent objects, but it can also be seen as synonym to instance. Handling these different meanings of the term requires different mental models.

RQ7 Which subject-specific strategies can we recommend?

We recommend educators and practitioners to establish and use a refined terminology for the notion of "object". A refined terminology will improve recognition of different mental models. Whether it will support computing professionals sufficiently in handling different mental models, is an issue that needs to be researched.

We recommend all computing curricula to explicitly cultivate awareness for possible different conceptualizations of abstract concepts.

RECOMMENDATIONS

We recommend all computing curricula to explicitly cultivate awareness for possible different conceptualizations of abstract concepts. Based on this study, we also recommend educators and practitioners to establish and use a refined terminology for the notion of "object". A refined terminology will improve recognition of different mental models.

FURTHER RESEARCH

As our research has limitations we suggest the following areas for future research:

PART II – Conclusions

- (1) The samples we were able to collect were small. The reliability of the scores of some the research questions was fair. To scaffold conclusions about the students' mental models, it is necessary to repeat the experiment.
- (2) We found indications for differences between groups of students, enrolled in different computing curricula. Education evolves, the IE curriculum does not exist anymore today, curricula Front End Development are emerging. It would be interesting to repeat the experiment in contemporary hybrid curricula.
- (3) An interesting question is, if the differences we have found result from different educational settings. Further research is needed to assess which differences are pre-existent to the enrollment in different computing programs.
- (4) Further research is needed to describe computing professionals' mental models, and investigate if the differences we describe are present in the computing practice.
- (5) How far is it possible to combine different abstract models during work operations? Combining models while discussing with colleagues might just be too complex for the human mind, even if the terminology is optimized.

Part III—Case Studies In A Hybrid Curriculum

RESEARCH QUESTIONS RQ5, RQ6, RQ7

RQ5 How do students in a hybrid curriculum experience a craftsmanship-based approach?

RQ6 Which subject-specific strategies were recommended in the past?

RQ7 Which subject-specific strategies can we recommend?

Authoring tools support users with little background in computing in the development of multimedia applications. These tools generally provide visual support for linking together pre-programmed elements, but they also allow customization through scripting. Although it is possible to develop an interactive multimedia application without writing code, coding becomes necessary to implement innovative features.

In 2008/2009, we investigated experiences of Multimedia students, who were exposed to a 3D Virtual World authoring tool, without being offered any support in how to conceptualize notions related to the software they were using. This approach was rather common at the time in Dutch Multimedia education, where computing was seen as instrumental and computing classes aimed at empowering students to concretize their ideas.

We had the opportunity to witness two editions of the same course. During the first edition, we explored the lecturers' assumption about the course's success. One of the course's assignments involved all students. They cooperatively created a village in a 3D Virtual World. According to the lecturers, the students had formed a community of learners during that assignment. This had resulted in greater engagement in the learning process, and better performances. In the second edition, we measured how students assessed their own learning experience. The hypothesis was, that increased sense of community during the assignment in the virtual world had triggered increased construction of learning. We were indeed able to measure an increased sense of community between the start and the end of the assignment. But despite that progression, despite the performance of the students, the students themselves reported that their learning had not increased during the hands-on assignment. They had learned in that time, but they had not learned more than in the weeks preceding the assignment.

7 A Craftsmanship-Based Approach (1)⁵

7.1 INTRODUCTION

3D virtual worlds have been around since the early nineties, originally, mainly in games and only in single-player mode. Later, multiplayer modes were added and because of increased use and growing internet bandwidth, the massive online role playing game mode was added to 3D games, like World of Warcraft (Vivendi, 2004). 3D is now no longer limited to games, it allows new ways of communication and even the possibility of having several lives, i.e. Second Life (Linden Lab, 2003). In this paper we focus on the use of 3D virtual worlds in an educational setting, and relate this to the course “Virtual Worlds” of the Hogeschool Utrecht University of Professional Education. Currently, innovating educational practice applying new technologies is an important topic (Educause Learning Initiative, 2006), though it is not straightforward that courses utilizing state-of-the-art technology have added value for students.

In this paper we explain the need for new paradigms when developing courses with new technologies. We illustrate this by positioning education in immersive virtual worlds in terms of the learning paradigm being applied. We suggest criteria for the successful implementation of a constructivist learning environment using a virtual world. We explore a few cases of the use of virtual worlds in education and identify some good and bad practices of using 3D virtual worlds. We then describe the “Virtual Worlds” course as a typical example of intertwining real life instructional education with a constructivist learning approach using a virtual world.

7.2 NEW TECHNOLOGY, NEW PARADIGM

New technology often changes the context of use, the constraints and the opportunities of application, making existing paradigms obsolete. This results in underutilization of new technology, until new paradigms arise and users become familiar with the new way of interaction.

Literature on education shows that the traditional instructional methods can be supplemented or even be replaced by constructivist learning

⁵ This work was originally published as: Benvenuti, L., Hennipman, E., Oppelaar, E. R., van der Veer, G. C., Cruisjberg, R., & Bakker, G. (2010). Experiencing and learning with 3D virtual worlds. In J. M. Spector, D. Ifenthaler, P. Isaías, & D. G. Kinshuk, *Learning and instruction in the digital age* (p. CH 12). NewYork, NY, USA. © 2010 Springer Science+Business Media, inc. doi: 10.1007/978-1-4419-1551-1_12

Computing Education in a Hybrid World

methods when using new technology, such as the upcoming and still evolving e-Learning environments and, in the last decades, immersive virtual worlds (Dede, 1995), (Dickey, 2005), (Educause Learning Initiative, 2006), (Antonacci & Modaress, 2008).

Eliëns, Feldberg, Konijn & Compter (Eliëns, Feldberg, Konijn, & Compter, 2007) describe applying the traditional learning paradigm to virtual worlds as “rather naive”: for the virtual campus of the Vrije Universiteit there was “frankly no reason to include what may be considered an outdated paradigm of learning” especially when “there might be more appealing alternatives”. However, other literature claims that there is added value in using virtual worlds preserving the traditional learning paradigm in a virtual classroom setting, mainly in creating a sense of a classroom community and in the fact that students will more easily join in class discussion in virtual life than in real life (Lamont, 2007), (Martinez, Martinez, & Warkentin, 2007), (Ritzema & Harris, 2008).

7.2.1 CONSTRUCTIVIST LEARNING

First of all, we will describe the notion of constructivism. The constructivist philosophy asserts that all knowledge is constructed as a result of cognitive processes within the human mind (Orey, 2001). We apply constructivism as a theory of learning: “Constructivism is (...) a *theory of learning* based on the idea that knowledge is constructed by the knower based on mental activity” (EduTech Wiki, sd). Constructivism knows different perspectives and induces various educational strategies. Most of them assert that the learning activity is supported by social interaction. The Constructionist strategy focuses on the interaction between the individual and the environment. According to constructivists, learning occurs through interaction and reflection; learners can create meaning by building artifacts (Orey, 2001). Notions which often are associated with constructivist learning are: collaborative problem solving, knowledge building communities, situated learning, experiential learning, immersive environments, participatory processes, interaction, learning by doing, activity theory, critical learners, as well as other terms.

We will define constructivist learning as the process of creating, sharing and evaluating knowledge, skills and understanding in a collaborative environment through interaction with that environment. The process results in the learner’s ownership of what is learned. We emphasize the learner’s responsibility for the acquisition and the management of knowledge, skills and understanding.

7.2.2 WHY 3D VIRTUAL WORLDS AFFORD CONSTRUCTIVIST LEARNING

Course management systems (CMSs) such as Blackboard, Moodle, and WebCT can be considered virtual learning environments (VLEs). These CMSs provide tools for creating virtual communities and are a central place where students meet, discuss their work, find and organize course materials, and discuss the course content with their lecturers. Though not very immersive, VLEs allow the emergence of knowledge-building communities, promote an interactive style of learning, have opportunities for collaboration and have meaningful engagement across time and space and thus enable constructivist learning (Dickey, 2005).

Current VLEs often enable more visual immersion, yet still providing the same tools and assets to education as the traditional CMSs have. These VLEs enable students to see each other and the lecturer by means of webcams, using for example Acrobat Connect (Adobe Systems, sd). Though still not fully immersive, these applications have a huge advantage over traditional CMSs in the fact that they allow for non-verbal communication and create a stronger sense of community.

Fully immersive massively multiplayer virtual worlds such as Active Worlds and Second Life have seen a rapid growth over the past decade (especially the last few years). These growing communities in virtual worlds with no preset narrative have spawn interest from both (large) companies, researchers, and educational institutes. We will focus on the researchers and educational institutes. In these worlds, learners themselves construct knowledge through interpreting, analyzing, discovering, acting, evaluating and problem solving in an immersive environment, rather than through traditional instruction (Antonacci & Modares, 2008). Especially virtual worlds with no preset narrative, such as Second Life and Active Worlds are considered to be a very usable asset in education (Livingstone & Kemp, 2006). These worlds differ from massively multiplayer online role-playing games (MMORPG) in the sense that there is no objective in the virtual world, other than social presence. Though we focus on virtual worlds without preset narrative, even virtual worlds that do have a preset objective (i.e. games) support learning (Steinkuehler, 2004) as well as research. The MMORPG World of Warcraft suffered from a corrupted blood epidemic, a situation that now is considered a disease model by some scientists (Gaming Today, 2007). Although learning is done in the virtual world, the skill and knowledge gained in virtual worlds is as real as it gets. This aspect of virtual learning is being researched to help people with Asperger's to socialize (Loftus, 2005) (Kirriemuir, 2008).

Computing Education in a Hybrid World

According to Dede, a virtual world needs at least two essential capabilities for constructivist education: (1) *telepresence* (via avatars) and (2) *immersion*, "the subjective impression that a user is participating in a "world" comprehensive and realistic enough to induce the willing suspension of disbelief" (Dede, 1995). But "immersion" is not an absolute quality: one can feel "somehow immersed" in a virtual community. Moreover, immersion does not only depend on the application. Participants in the first newsgroups felt probably more "immersed" in the Internet-community than they would feel now if we were using the same applications.

We can say that the capabilities Dede describes are less prominent in traditional CMSs (or VLEs) than in 3D virtual worlds. Present-day's students would probably not even consider CMSs as 'virtual environments'. We therefore assume that constructivist learning is better supported by 3D virtual worlds, than it is by traditional CMSs. This is something a lecturer should realize when considering new technologies for a course.

7.2.3 DOWNSIDES OF CONSTRUCTIVIST LEARNING

It is difficult or sometimes even impossible to define learning goals in a constructivist learning setting (Educause Learning Initiative, 2006) (Jonassen & Roher-Murphy, 1999). This automatically results in difficulty of assessment; with no fixed learning objectives there is no easy way to assess whether objectives have been met. Jonassen and Rohrer-Murphy (Jonassen & Roher-Murphy, 1999) argue that: "designers committed to designing and implementing CLEs (Constructivist Learning Environments) need an appropriate set of design method for analyzing learning outcomes and designing CLEs that are consistent with the fundamental assumptions of those environments". They propose the use of an activity-theory based framework to assist in the mentioned tasks, because activity theory closely relates to constructivist theory on collaborative problem solving, experiential learning (Mason, 2007) and situated learning (Hayes, 2006).

Immersive virtual worlds have another downside as well. Being 'immersive', they can be so engaging to students that it distracts them from the actual course (Educause Learning Initiative, 2006). A good example was found by Martinez et al. (Martinez, Martinez, & Warkentin, 2007) where a student did not come to the course because he rather spent his time in a virtual bar.

7.2.4 DO 3D VIRTUAL WORLDS ALWAYS SUPPORT CONSTRUCTIVIST LEARNING?

We agree with Dede (Dede, 1995) that 3D virtual worlds can support constructivist learning, but in our view, the success of a constructivist virtual learning environment does not depend on “telepresence” and “immersion”. We will provide one example of an educational application in a virtual world allowing telepresence and immersion, but with no added value for the learners. We will also discuss successful learning environments using 3D virtual worlds and will discuss key features for the implementation of similar environments.

We are looking for criteria for the successful implementation of constructivist learning environments with virtual worlds. Some key features for education that are supported by virtual worlds have been described by Cross, O'Driscoll and Trondsen (Cross, O'Driscoll, & Trondsen, 2007) and related to learning strategies: (1) Flow - balancing challenge and inactivity for an engaging experience; (2) Repetition - virtual worlds allow for endless repetitions with no extra costs; (3) Experimentation - virtual worlds allow for simulation and modeling; (4) Experience - being part of a collaborative community; (5) Doing - virtual worlds are big practice fields; (6) Observing - learning by observing how others do things; (7) Motivation - the rich context motivates through situated learning. We agree with the criteria, but unlike Cross, O'Driscoll and Trondsen we do not think that learning exclusively takes place in the virtual world. We will paraphrase those criteria, mainly by renaming them, and will add (8): Reflection, as a key feature of constructivist learning that does not necessarily take place in the virtual world. This way, we will focus on: (1) Flow; (2) Training; (3) Experimentation; (4) Collaboration; (5) Learning by Doing; (6) Observing; (7) Motivation and (8) Reflection.

7.2.5 HOW TO ASSESS SUBJECTIVE IMPRESSIONS

We identify (1) Flow; (2) Training; (3) Experimentation; (4) Collaboration; (5) Learning by Doing; (6) Observing; (7) Motivation and (8) Reflection as key features for constructivist learning environments with virtual worlds.

Some of those criteria are objective. Virtual worlds always allow Training and Experimentation. The fulfilling of some of the other criteria depends on the situation. Learning by Doing, Observing and Reflection can be supported by the educational setting. If the circumstances are known, it is possible to check whether those criteria are met or not. “Flow,

Computing Education in a Hybrid World

Collaboration and Motivation, however, are subjective criteria. They apply to the learners' experience in interaction with the environment rather than to the learning environment itself. To validate those criteria, we should measure the learners' experience.

7.2.6 MEASURING EXPERIENCE

In our view, experience is not an isolated phenomenon, but a process in time, in which individuals interact with situations. During this process interpretation takes place: individual meaning is constructed. Therefore we consider experience a subjective and constructive phenomenon. Both the individual and the situation contribute to the experience, that's why we argue for a holistic approach in measuring experience. Measuring experience should include assessment of the respondent's expectations, of the actual "living through" the experience and of the after effects (Vyas & van der Veer, 2006).

Vyas and van der Veer (van der Heide, 2002) developed several strategies to assess the respondent's interpretation of the situation at different moments in time. Basically, they interview the respondents prior to the experience and afterwards. The assessment of the "living through" is done by observing the respondents who perform tasks while they talk aloud.

Another tool which is used to measure subjective impressions is the Visual Analogue Scale. Visual Analogue Scales or VAS scales are used in the medical world for subjective magnitude estimation, mainly for pain rating. They consist originally of straight lines of 10.0 cm long, whose limits carry verbal descriptions of the extremes of what is being evaluated. VAS scales are of value when looking at changes within individuals; their application for measuring a group of individuals at one point is controversial (van der Heide, 2002) (Langley & Sheppard, 1985).

7.3 OBSERVATIONS IN THEORY AND PRACTICE

3D virtual worlds enable modeling and simulation, and they facilitate communication between personalized avatars. In some cases intelligent agents can be created for the virtual world. These aspects result in the creation of virtual lives and the formation of social networks; in many ways similar to those in reality. This resemblance of reality makes these virtual worlds an immense virtual 'lab' for communication studies, social studies, psychology studies (Stanford University, 2001), architecture, and medical studies (Kamel Boulos, Hetherington, & Wheeler, 2007). These and many other examples of educational use of Second Life have been collected by

A Craftsmanship-Based Approach (1)

Conklin (Conklin, 2007). From a broad range of examples, we have chosen two that in our opinion are exemplary for constructivist learning in virtual worlds.

Our first example is the VNEC (Virtual Neurologic Education Centre) on Second Life. VNEC was developed by Lee Hetherington at the University of Plymouth, Devon, in the United Kingdom. It contains a simulation where people (avatars to be more precise) can experience common symptoms that may occur from a neurological disability (Kamel Boulos, Hetherington, & Wheeler, 2007). This is an immersive experience, partly to make people aware of neurological disabilities, but the education center has information facilities as well. In terms of the criteria mentioned in section 7.2, this education center scores very well on all eight points. A member group is associated with the VNEC, so a community has been formed around it. VNEC uses the available technology to near-full potential, and it creates an environment that has most elements for constructivist learning.

The second example is an experimental course by Polvinen (Polvinen, 2007) for Fashion Technology students at Buffalo State College. This course consisted of both a real-world part and a virtual-world part, where the virtual-world part complemented the real-world. The students used Second Life to create fashion, doing a fashion show, creating fashion collections with a vendor display and many other activities that fashion designers would do in real life. Polvinen concluded that “all the aspects involved in real world production of a fashion show can be simulated in the virtual world as well as fashion product design, development, and presentation”. In this course virtual worlds are a very cost-effective way to simulate processes from the real world, including many social, business and psychological aspects.

Multimedia Virtual Worlds are especially useful in constructivist learning, by enabling experiential learning, situated learning, and collaborative problem-solving. From existing examples of education using virtual worlds and from literature, we draw the following conclusions.

Virtual classrooms in a traditional instructional setting do not fully utilize the possibilities of the immersive virtual world. They do however have added value over non-immersive virtual worlds by creating a ‘class’-feeling. Compared to real-world classroom settings, the virtual ones suffer from the engaging context of the classroom (virtual worlds encourage exploration). An advantage of the virtual classroom over real-world classrooms is that students join a discussion more easily through chat, than they would do in real life. Whether this is an advantage is debatable: do students acquire the competence to speak in public?

Computing Education in a Hybrid World

When transforming a course from real-life into a virtual world, the initial learning objectives of the course will need to be reformulated and assessment should be reconsidered. Virtual worlds afford another type of learning than real life education. The learning objectives of courses in virtual worlds should match the possibilities of a virtual approach, and assessment should take place in an appropriate way.

Modeling, simulation, and collaboration are effective tools for knowledge creation and knowledge transfer. In general, you can get rid of real world constraints (Wages, Grünvogel, & Grützmacher, 2004).

Virtual worlds are especially an asset to real life education when students can try out concepts that would be too difficult, too expensive, or too risky in real-life, or when lecturers need to demonstrate things that are difficult, if not impossible, in real-life, such as complex large-scale molecule models.

The course “Virtual Worlds” (which will be described below) combines the best of both worlds. Traditional instructional methods (classroom setting) in real life provide the course framework, a solid knowledge base, and create opportunity to share design knowledge. The virtual world (in this case an Active Worlds world) is used for practical assignments. We found that this complementary approach works very well, collaboration continued naturally in both worlds (virtual and real), resulting in a ‘community of learners’.

7.4 A COURSE ON 3D VIRTUAL WORLDS

The course “Virtual Worlds” is part of the Digital Communication curriculum (third or fourth year of University College) of the Hogeschool Utrecht University of Professional Education in the Netherlands. The goal is to teach students to think about virtual worlds in a conceptual way. Because of the curriculum, the course deals with virtual worlds from a communication perspective: the exercises are about possible usage of virtual worlds, advantages and disadvantages.

One half of the course is about theory, the other half is about practice. Contrary to previous years, in the spring 2008 course, students’ practical assignments took place in a shared virtual world (enabling collaboration), situated in Active Worlds. The students were asked to build contributions to an ‘Asterix village’. Assessment and discussion took place in a plenary closing meeting. Active Worlds is a virtual world platform without preset narrative. This enabled students to individually develop and implement interaction concepts (in the current course a house and some activity in or

A Craftsmanship-Based Approach (1)

near the house) as well as to collaboratively develop interaction concepts for the entire village.

7.4.1 COURSE STRUCTURE

The 'Virtual World'-course takes seven weeks. It starts with an introduction to design concepts within virtual worlds and on how to design an experience. In the first part of the course, theory focuses on perceptual opportunities. Best practices are considered. A lecture discusses whether rendering photorealistic 3D or other techniques may increase credibility of virtual worlds (Bakker, Meulenberg, & de Rode, 2003). In this part the students practice building a VR application, (a model of a house and its environment for the Asterix-village) in 3D Studio Max.

In the second part of the course, theory focuses on the future of virtual worlds when artificial intelligence and photorealistic rendering will increase opportunities for agent intelligence and for interaction within virtual worlds. The practical part consists of an assignment to bring their little virtual world concept to life, and to describe an interesting application for the common virtual world, the Asterix village. Examples of applications: illustrating how ancient people constructed their homes; and a "language village" for visiting avatars speaking a foreign language.

In the final weeks students reflect on the benefits of virtual worlds and present their work.

7.4.2 A VIRTUAL WORLD AS AN EDUCATIONAL TOOL

Building a village collaboratively promised advantages over individual projects. Students had to learn how to implement models in Active Worlds from each other, especially since no tutorial was available. In fact, by working as a group they all succeeded to take this hurdle relatively fast. Their houses and surroundings would never have reached the sophisticated level if they had not been able to learn and copy from each other and from other virtual worlds. In addition, creating a real virtual world as a group is expected to stimulate and motivate students as the result, for each individual, is really a new world to explore.

Re-using each others' work raised unexpected issues: one student created an animated smoke that was soon featuring the chimneys of many others. The teachers, needing to establish individual students' credits, easily established who was the original creator of unique features, and successfully discussed the concept of "intellectual property" as a side effect.

Computing Education in a Hybrid World

They also established a citation index, to guarantee that the creators of interesting features would benefit from their work.

7.4.2.1 What does not work

Without an adequate strategy, a virtual world is just a playground. In a previous course, a Bachelor student presented his thesis work (on the possibilities of Active Worlds for schools and universities). His assignment included the creation of a building with classrooms in the virtual world and required him to present in one of those classrooms, where “presence” for all students was compulsory. However, the students’ avatars did not have any role other than to sit and watch the presenting avatar. In this situation, the use of a virtual world did not show any advantage. Students’ avatars were indeed in the virtual room but, instead of participating in discussion, were engaged in activities like dancing the Macarena, labeled by abstract nicknames.

7.5 ASSESSMENT

The ‘Virtual Worlds’ course was not an experiment. We heard about the course when it was about to reach its conclusion. The teachers reported interesting issues:

- The course was very successful: all the students accomplished their assignment, compared to 33% needing a second assignment in the past;
- The application of new technology was challenging. No tutorial of Active Worlds was available; conversion from 3D Studio Max was problematic and implementing interactivity reduced the performance seriously. Despite of all this, the performance of the course increased dramatically compared to previous editions;
- The Hogeschool Utrecht always performs student evaluations. The Virtual Worlds course was rated 8.4 on a scale from 1 to 10, which is the highest score ever.

Did the ‘Virtual Worlds’- course meet the criteria for the implementation of a successful constructivist learning environment that we described in section 7.2.4?

Obviously, implementing a 3D-application is an activity which allows Training, Experimentation and Learning by Doing.

Since the results were placed in a common virtual world and sharing sources was allowed, the students could Observe fellow-students who were

A Craftsmanship-Based Approach (1)

performing the same task and give them feedback. Reflection was supported this way and encouraged during classroom meetings.

Assessing Flow, Collaboration and Motivation is more complicated. Those questions concern the students' experience.

7.5.1 THE SURVEY

To assess the students' experience we might have performed interviews before the course, have followed the students during the course and have interviewed them again after conclusion. Since our involvement with the course started late, the best possibility to perform the assessment was to develop one questionnaire with 3 search areas:

The students' expectations prior to the experience

- Q1. What were your expectations when you started with the 'Virtual Worlds' course?
- Q2. What did you think you were going to learn in this course?

The actual living through

- Q3. What is the most important thing you have learned by cooperating with the class in a virtual world?
- Q4. Did you witness things you had never heard of during this course?
- Q5. Did anything surprise you during this course?

The after effects

- Q6. What do you tell your friends and family about this course?
- Q7. What would you advise the school on the next implementation of this course?

Additionally, we asked the students to express themselves on issues related to their attitude, their view on the usefulness of cooperation and on the way the course was taught. We asked them to rate their answers to those questions on Visual Analogue Scales (4.1). For obvious reasons, we performed only one measurement. We asked the students to rate their answers to the following questions:

Their attitude during the course

- V1. Passive / Active
- V2. Not involved / Very involved

Computing Education in a Hybrid World

Their opinion about the course

V3. Boring / Surprising

V4. Unattractive / Attractive)

Their opinion about the cooperation in the virtual world

V5. Not instructive /Very instructive

V6. Not useful / Useful

7.5.2 RESULTS

At the time we performed our survey, the course was finished. We managed to locate 14 students out of 19; they all responded to our request.

Their answers to the questionnaire showed:

- Issues concerning the students' expectations prior to the experience (Q1, Q2)

The students started with different expectations about what they would learn (3D modelling: 5; applications of Virtual Worlds: 5; 3D modelling and Virtual Worlds: 4).

- Issues concerning the living through the experience (Q3, Q4, Q5)

The majority of the students had been positively surprised by the course (Yes: 10, Not surprised: 3). Most of the students had witnessed things they had never heard of (Yes: 10; No: 3, not answered: 1). One student was negatively surprised by the amount of time 3D modelling requires but reported nevertheless positively to friends and family.

- Issues concerning the after effects (Q6, Q7):

Technology was challenging: many suggestions for improvement concerned this subject (4 were related to Active Worlds, 5 to 3D Studio Max, though 5 suggested to not change anything).

10 students looked back at the course with explicit satisfaction ("I worked very hard but had a lot of fun", "it was fun to do and we could work together in the virtual world", "It was worth it") despite of the technical problems they mentioned. Almost everybody who did report to friends and family had reported positively. The only respondent who had reported negatively, had thought he was going to learn how to make a complete 3D environment and had been disappointed by the course mainly discussing

A Craftsmanship-Based Approach (1)

design theory and the state of 3D-technology. He would rather have spent more time in class discussing technical issues.

For the second part of the survey, we used VAS scales. VAS scales are mostly used to assess changes in time of individual ratings of subjective findings: comparing the results of a group of individuals is not straight forward. We are aware of this problem, which is why we only give an impression of the results. The questions concerned:

- The students' attitude (V1, V2):

How was your own attitude during the course?

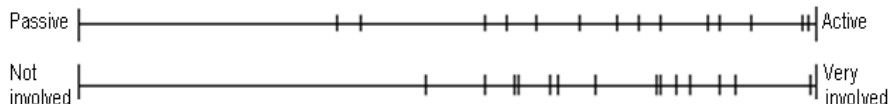


Figure 7-1 The students' answers to the question concerning their own attitude

- The course (V3, V4)

What do you think of the course?

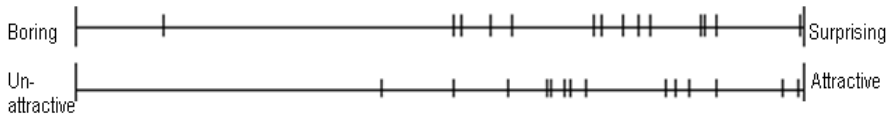


Figure 7-2 The students' opinion about the course

- The cooperation in the virtual world (V5, V6):

What do you think about cooperating in a virtual world as a class?

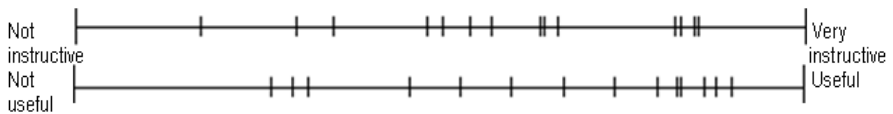


Figure 7-3 The students' opinion about the cooperation

To summarize:

- Figure 7-1 shows that most of the students called their own attitude "active"; almost everybody considered themselves involved.

Computing Education in a Hybrid World

- Figure 7-2: The large majority of the students reported having been surprised by the course. The students labelled the course “Attractive”.
- Cooperation was not always evaluated positively (Figure 7-3). This might be related to the problems the group had encountered with intellectual property. One of the students who had criticized cooperation answered question Q3 saying that he had learned that “they steal everything. My fishes!”. Nevertheless, even at this point the scores were mostly positive.

Although we did not prove that the learning environment of the ‘Virtual Worlds’ course meets all the criteria we identified in section 7.4.2, the results of our survey support a positive answer to our question.

7.6 CONCLUSIONS AND FUTURE WORKS

Current literature about ‘best practices’ and our own findings in our Active Worlds course show the following:

- In 3D virtual worlds, it is possible to show (and teach about) some things in a more realistic way than in a real world since, as Wages, Grünvogel and Grützmaker (Wages, Grünvogel, & Grützmaker, 2004) state: one can get rid of real world constraints.
- In a shared virtual world students can learn from each other and cooperate in creating, which, in addition, is strongly motivating.
- Individual assessment is not straight forward as students work in groups. Copying and reusing each others’ work should in many cases be considered positive learning behavior, though special measures are needed to credit individuals for unique contributions. Discussing this in (real) group sessions works well, as do individual plenary final presentations.
- Moving traditional classroom activities into virtual worlds is not always effective, as in the case of attending lectures.

There will be new editions of the ‘Virtual World’ course. We intend to investigate the students’ experience, to scaffold our claim that this course meets our criteria for a constructivist learning environment with a virtual world.

The 2008 “Virtual Worlds” course was successful and an improvement to its classical predecessors. Students were more present, more active, and, according to informal evaluation, more satisfied. Teachers estimate that

A Craftsmanship-Based Approach (1)

students gained more conceptual insights in virtual world applications. This success surpassed teachers' expectations.

It is evident that the relation between the course subject and the applied technology was fundamental: a course about virtual worlds with an assignment in a virtual world is a powerful combination. Other success factors might be the choice for an immersive collaborative structure, or the village metaphor.

Before we would dare to generalize this local success, we intend to explore other learning domains, and other collaborative narratives and related world metaphors.

ACKNOWLEDGEMENT

We would like to thank the students of the course "Virtual Worlds".

8 A Craftsmanship-Based Approach (2)⁶

8.1 INTRODUCTION

The Digital Communication curriculum of the Hogeschool Utrecht (the Netherlands) has, as one of its courses, a course on Virtual Worlds. As a strategic choice, the lecturers decided to rely on the same technology for the implementation of the learning and teaching environment.

New technologies in classroom offer opportunities for innovation. They also change the way students experience education. This can be challenging for educators who are confronted with unexpected situations. But use also promotes understanding. Through an assignment in a virtual world the students would gain more familiarity with the technology. The lecturers would get better acquainted with the didactic possibilities of the medium.

In this paper we share our observations from practice. We focus on the educational possibilities of virtual worlds with no preset narrative by describing the Virtual Worlds course, which partially took place in an Active Worlds environment. We discuss the advantages of moving part of the action to a virtual setting, but also point at its limitations. We describe some problems that were raised by the medium and offer strategies to deal with them.

We show how to improve our understanding of new media and technology by combining education with research. In particular we measured the students' experienced "connectedness" and "learning" in the time span of the course.

8.2 A VIRTUAL VILLAGE AS A COMMUNITY OF LEARNERS

Virtual worlds with no preset narrative, such as Active Worlds or Second Life, are considered usable asset in education (Livingstone & Kemp, 2006). The implementation of the learning environment in a virtual world is not necessarily successful.

The "Virtual Worlds" course of the Digital Communication curriculum of the Hogeschool Utrecht (University of Applied Science, Utrecht, the Netherlands) combines traditional instructional methods (classroom setting) with a collective practical assignment in an Active Worlds world.

⁶This work was originally published as: Benvenuti, L., & van der Veer, G. (2011). Practice what you preach: experiences with teaching 3D concepts in a virtual world. In S. H.-J. (ed.), *Virtual Immersive and 3D Learning Spaces: Emerging Technologies and Trends* (p. Ch. 3). ©2011 IGI-global, doi: 10.4018/978-1-61692-825-4.ch003

Computing Education in a Hybrid World

Starting with the spring of 2008, the students are asked to build an application, consisting of 3D-model of a house and its surroundings, and to place it in a common village situated in an Active Worlds environment. Figure 8-1, as well as Figure 8-4 and 8-5, show examples of students' work during the courses we analyzed.

The complementary approach of the Virtual Worlds course works very well: collaboration continues naturally in both worlds, virtual and real (Benvenuti L. , et al., 2008). The spring 2008 edition of the course was very successful. The relation between the courses subject and the applied technology was a powerful combination. Other success factors might be the choice for an immersive collaborative structure, or the village metaphor.

The question whether a virtual village is an appropriate tool to support a community of learners is not one-dimensional. Sense of community is a subjective concept. Some students consider participating in a community equivalent to sharing results with colleagues; others expect the results to be produced together.

The same applies to learning: the students who pass an exam frequently don't agree on what they have learned, nor on its significance. In 2008, we had investigated the students' opinion about the way the course was taught and about their own attitude. Though the answers were obviously positive – the students stated having found themselves involved and participating in the learning community - we noticed a large variety of opinions on the usefulness and instructiveness of the collaboration in the virtual world.



Figure 8-1 The spring 2008 Asterix village: house of the bard and surroundings. ©2008 Hogeschool Utrecht, used with permission

A Craftsmanship-Based Approach (2)

With the introduction of the assignment in a common virtual world, the course's results had increased beyond the lecturers' expectations. In their opinion, this was due to the fact that the students had been able to learn from each other more than in the previous editions of the course. The students themselves seemed more cautious in sharing that conclusion.

In 2009, we had again the opportunity to perform measurements in the spring edition of the Virtual Worlds course. We decided to focus on the two issues mentioned above: experienced community membership and experienced learning.

8.3 MEASURING EXPERIENCED CONNECTEDNESS AND LEARNING

In our view, experience is not an isolated phenomenon, but a process in time, in which individuals interact with situations. During this process interpretation takes place: individual meaning is constructed. Therefore we consider experience a subjective and constructive phenomenon. Measuring experience should include assessment at three moments in time: the respondent's expectations before the experience, their assessment during the experience and afterwards (Vyas & van der Veer, 2006).

Vyas and van der Veer developed several strategies to assess the respondent's interpretation of the situation at different moments in time. Basically, they interview the respondents prior to the experience and afterwards. The assessment of the "living through" is done by observing the respondents who perform tasks while they talk aloud. This was not possible in a classroom setting, so we decided to assess the students' subjective impressions by conducting the same written survey at three moments in time.

A reliable tool for measuring subjective impressions is the Visual Analogue Scale. Visual Analogue Scales or VAS scales are used in the medical world for subjective magnitude estimation, mainly for pain rating. They consist of straight lines of 10.0 cm long, whose limits carry verbal descriptions of the extremes of what is being evaluated. Because the interval between the limits does not carry any verbal label, the exact location of the value that respondents mark will not be remembered, so with repeated measurements it is impossible that respondents just repeat their previous scores. VAS scales are of value when looking at changes in time within individuals (Langley & Sheppard, 1985).

The impressions we were interested in are the student's sense of belonging to a community and the extent to which student's learning goals

Computing Education in a Hybrid World

are met. The Classroom Community Scale or CCS (Rovai, 2002) was a good starting point. The CCS is a psychometric scale, developed using factor analysis with 2 principal components: Connectedness and Learning. Connectedness indicates the perceived cohesion, spirit, trust and interdependence between members of the classroom community. Learning represents their feelings regarding interaction with each other as they pursue the construction of understanding. Learning also indicates the degree to which they share values and beliefs concerning the extent to which their educational goals and expectations are being satisfied.

Our variant of the Classroom Community Scale applied VAS-scales instead of Likert scales because this prohibits memory of previous scores when applied to repeated measures.

8.4 THE ASTERIX VILLAGE

The “Virtual Worlds”- course (3rd year in the curriculum) takes seven weeks. The course’s goal is to teach students to think about virtual worlds in a conceptual way. The students should learn to establish when it is appropriate to make an application in a virtual world. Virtual worlds are powerful instruments, but developing a VR application can be very expensive. Sometimes a less demanding solution is better.

Half of the course is dedicated to theory, the other half is practical. Theory is assessed individually, with a written test. The course has two related practical assignments; in the laboratory the students work in pairs.



Figure 8-2 General course outline

The course starts with an introduction to virtual worlds. Design concepts within virtual worlds are discussed. In this part, students develop a VR model of a house and its environment in 3D Studio Max. In order to stimulate relations between individual designs, the lecturers introduced a common theme. In 2008 and 2009 the common theme was “Asterix”.

A Craftsmanship-Based Approach (2)

In the second part, theory focuses at the future of virtual worlds when artificial intelligence and photorealistic rendering increases. Students are also stimulated to find best practices. The focus switches to the second practical assignment which takes place in a shared Active Worlds context, the Asterix village. Active Worlds has no preset narrative; this triggers students to develop and implement interaction concepts individually as well as to develop these collaboratively for the entire village. The setting –a village– was chosen consciously, to obtain a common structure.

In the final weeks students reflect on the benefits of virtual worlds and of virtual worlds in cross-media concepts. At the end of the course students present their work in a plenary classroom meeting. Discussion and assessment of the practicum assignment also takes place in that meeting.

8.5 WHAT'S NEW, WHAT'S NEXT?

The spring 2008-edition of the Virtual Worlds-course was not the first one. The upper part of the course outline in Figure 8-2 also applies to previous editions, but those courses only had one practicum assignment: design and develop a new VR-concept. In 2008, the assignment in the shared virtual world was added. Nothing else changed. The teaching timetables remained unaltered. The class meets twice a week in the multimedia laboratory. Meetings last 2 hours. The first hour is dedicated to theory, during the rest of the time the students work on their assignments, in pairs. Students are – and always were - stimulated to ask questions to the lecturers and to each other. Students are – and always were - stimulated to learn from each other.

Until the spring of 2008, students mainly exchanged information during laboratory time. If the lecturers were busy and a student had a well-defined question, it came naturally to ask other students. With the introduction of the assignment in the Active World world this pattern changed. Now students also “met” while working in the virtual world from home, in the evening or at night. As we can see in Figure 8-3, Active Worlds has a chat box. Communication comes easily. The first improvement which came with Active Worlds lays in the increased possibilities for students to discuss issues while they are working, and to support each other on the spot without face to face contact.

This is a powerful feature. In the first edition of the Virtual Worlds-course in Active Worlds, implementing 3DS Max models into Active Worlds was challenging, since good tutorials had not been written yet. But the

Computing Education in a Hybrid World

students succeeded to import their 3D-applications relatively fast by working as a group: one of them was very skilled in 3D developing and shared his knowledge generously. In the next editions of the course, the group had tutorials but lacked the experienced colleague. Technical problems seemed more difficult to overcome in that situation. In the spring of 2009, the lecturers added an extra meeting and invited an expert in 3DS Max and Active World to answer to the students' questions.

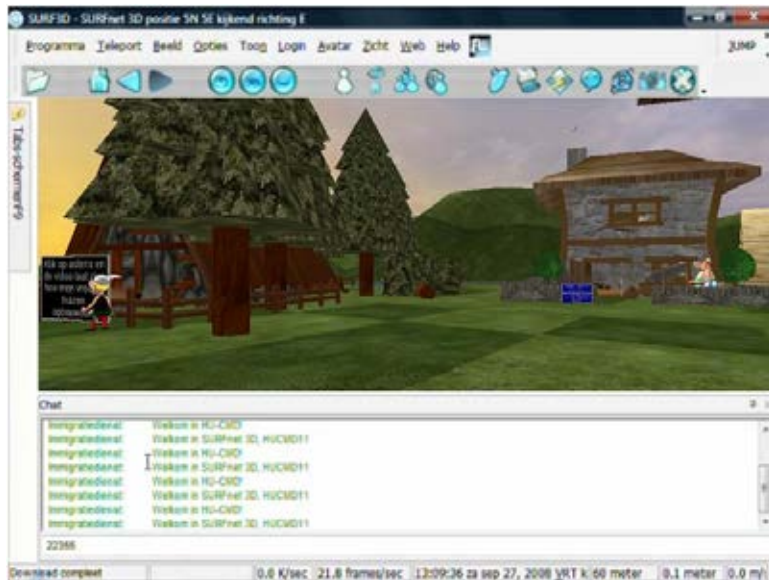


Figure 8-3 Active Worlds: the students' perspective with the chat box

The second improvement concerns the possibility to follow each others' progresses. Before the introduction of the common virtual village, students only discussed each others' work thoroughly during the conclusive meeting. These discussions always concerned the final versions of the products. While working in Active Worlds, students can follow the process of creation of all products. Every prototype is published and is visible for everybody. This triggers a fertile competition between the students: everybody wants his application to excel.

Software can be copied. The participants in the virtual village work in an open source environment: they can see and even copy parts of other students' work. This is very instructive and inspiring but also challenging. In 2008, one student had created animated smoke; a few days later most of the

A Craftsmanship-Based Approach (2)

huts in the village had smoking chimneys (see Figure 8-4). This was disappointing for the student who had invested time in developing what he thought would be a unique feature. He felt robbed.

The lecturers wanted to encourage the students to share results, but they also wanted to grant the credits for interesting ideas to the right persons. An appropriate assessment policy had to be developed. ‘Intellectual property’ was discussed in classroom and a citation index was established. From that moment, copycat behaviour was not seen as “stealing” anymore but as a tribute to the conceiver of the particular feature.

The third improvement lies in the possibilities to share results, to learn and copy from each other.



Figure 8-4 The spring 2008 Asterix village with the smoking chimneys. ©2008 Hogeschool Utrecht; Used with permission.

Of course, the question was raised if it was appropriate to skip the traditional meetings and move the whole course to the virtual environment. The answer is negative. The traditional classroom setting offers opportunities to plenary discussion. This is difficult to reproduce in a virtual environment, while it appears to be very useful. The problems concerning the intellectual property were discussed in classroom. Without the presence of all the students and their participation at the discussion it would not have been possible to come to a solution.

Presence is a cloudy concept in a virtual setting. Benvenuti et al (2008) discuss the case of a compulsory meeting in a virtual classroom, where the students had labelled their avatars with nicknames and made them

Computing Education in a Hybrid World

misbehave. But even if the avatar is present and respectful, it is difficult to know if the student is.

Not everything turned out to be feasible. After the first edition, the lecturers had planned to store the “best practices” so that they could be re-used the next year. This way they wanted to establish a canon of interesting, inspiring applications made by students. At the same time, they wanted to prevent new groups to start with an empty ‘world’. This turned out to be too ambitious. The best applications needed too much memory, therefore the system slowed down too much. This idea was frozen until technology will support it.

In order to overcome the “empty world” feeling, the lecturers instructed one group of students to provide for different strategies to induce the experience of being part of a village structure. They did so by implementing a palisade to demarcate the border, a map of the village to support navigation and several means of transport to encourage exploration. An impression of those solutions is given in Figure 8-5.



Figure 8-5 Strategies to delimit the action. ©2009 Hogeschool Utrecht; Used with permission

8.6 PERCEIVED CONNECTEDNESS AND LEARNING

Our research goal was to investigate the students’ experience on community membership and learning. First of all, we have to emphasize that conducting research in real-live educational setting is difficulty. We had the opportunity to witness two editions of the same course in two consecutive

A Craftsmanship-Based Approach (2)

years, which is promising. In 2009 we collected data. But a course is not an experiment; there are many important variables we were not able to control, first of all, the composition of the group. But also the moment in time in the school's history, with the school starting a new program, and the lecturers' intermitted absence (because of illness and other personal circumstances) probably have had an impact on the way the course evolved.

Measuring experience included assessment of expectations prior to the experience, during the experience, and after the facts (Vyas & van der Veer, 2006). We measured the students' perception of belonging to a classroom community and their perception on accomplishing (subjective) learning goals. We used the questions of the Classroom Community Scale (Rovai, 2002), structured along the Visual Analogue Scales (VAS) to identify changes within individuals (Langley & Sheppard, 1985). These questions consist of 2 domains: Connectedness (10 questions) and Learning (10 questions) that are scored on an analogue scale from 0-10. We had to delete one item of the learning questionnaire because it was poly- interpretable to our Dutch audience.

In 2009, 15 students followed the Virtual Worlds-course. The survey was filled in the first meeting (week 1), again in the meeting where the students gained access to Active Worlds (week 4) and finally in the conclusive plenary meeting (week 8). 5 students were absent in one of more of those meetings, so our findings concern the 10 students who filled all the surveys.

We calculated the initial change in connectedness and in learning by subtracting the scale scores of week 4 from week 1, and calculated the later change by subtracting those of week 8 from week 4 for each individual. We calculated the significance of the change scores by applying the student t-test against the 0-hypothesis of no change in the population.

We found that the sensed/perceived "connectedness" increased significantly (Student t-test, $p < 0.05$) in the first part of the course (between the start in week 1 and week 4, when students meet each other and start collaborating) as well as in the second part (from week 4 to week 8, when the collaboration is continued in the virtual world). The sensed/perceived "learning" only grew significantly in the beginning of the course (between week 1 and week 4) even though new experiences as well as new designed artifacts were in fact (objectively) evident from the students' behavior in Active Worlds.

We expected connectedness to increase after the common virtual world was introduced, despite of the fact that by then the students were acquainted to each other. But the lack of increase of the perceived learning puzzles us. It is possible that the setting triggers ambitions that are difficult

Computing Education in a Hybrid World

to fulfill. The software might be easier to learn than to master, especially if none of the community members has real expertise in 3D-programming. This can be an interesting starting point for further investigations.

8.7 CONCLUSIONS

The application of the Active Worlds environment to the Virtual Worlds course was a very fruitful step, to the students, the lecturers and the researchers.

The students gained insight into the possibilities of virtual applications, even if their satisfaction on their own learning leveled off in the last weeks.

The lecturers discovered new features – and new problems - of virtual worlds. They developed educational strategies on how to optimize the use of a virtual world as an educational tool and how to cope with the problems it entails.

The researchers had the opportunity to test their hypotheses on the effect of the application of this new technology in education, and to formulate new research goals.

We summarize our findings.

Common assignments in a Virtual World lead to:

- Increased possibilities for students to support each other on the ‘virtual’ spot. This is traditionally an advantage of face2face meetings. Now, at any time (and from any location) students turned out to be able to discuss with their colleagues online.
- The possibility to follow each other’s progress, which is inspiring and stimulating.
- The possibilities to share results, to learn and copy from each other in an ‘open source’ setting.
- A stronger sense of community. The perceived sense of connectedness, of belonging to a community, increases when the students collaborate in a virtual world as a group.

Traditional classroom meetings are still useful:

- To support plenary discussions.
- To capitalize the historical knowledge of classroom teaching. The lecturers intended to “reuse” the best practices by keeping them in the virtual village during the following editions of the course, to establish a canon of inspiring applications. Current technology does not (yet) support this option.

A Craftsmanship-Based Approach (2)

We found new questions for further discussion and research:

- Allow students to collaborate and even to copy from each other asks for a re-design of the assessment policy.
- Why does the perceived learning level?

Our recommendation is: practice what you preach in classroom and use the opportunities this combination offers to learn. Even if the topic of discussion is a moving target, indeed, precisely in that case it is important to rely on one's own experience to draw one's own conclusions.

ACKNOWLEDGMENTS

We thank the lecturers of the Virtual Worlds course, Bob Cruijsberg and Geeske Bakker, for having admitted us in their classroom. We also thank the students of both courses for their cooperation and their products.

Part III - Conclusions

We have investigated experiences of Multimedia students, who were exposed to advanced technology (3D Virtual World authoring tools), without being offered any support in how to conceptualize notions related to the software they were using. This approach was common at the time in Dutch Multimedia education.

RESEARCH QUESTION RQ5

RQ5 How do students in a hybrid curriculum experience a craftsmanship-based approach?

In the Virtual World course, the students' accomplishments surpassed the lecturers' expectations. The lecturers attributed this success to the formation of a community of learners during the hands-on assignment in the virtual world. We partially agree with them. The learning objectives concerned thinking about virtual worlds in a conceptual way. The assessment policy was aligned with the course's educational ambitions. The lecturers were indeed pleased by the results. But at the end of the 2009 edition of the course, the students reported that their learning had not increased during the assignment in the virtual world.

To support each other in their learning process, the members of a learning community should agree with each other (and with their lecturers) upon the objectives that are pursued. We think that the students' learning ambitions during the assignment in the 3D Virtual World did not match the lecturers' educational ambitions. From the lecturers' perspective, the course showed a consistent picture. But the challenges students had to face during the assignment were not aligned with the learning objectives. During the assignment, students had to solve problems that did not concern the concepts behind virtual worlds, but the implementation of 3D applications in a specific Virtual World. Most students had overcome these problems somehow, and the sense of community had increased in the 2009 edition of the course, but no increase of the students' learning experience was measured in the 2009 class.

Our interpretation of the students' responses is that when they had stated that their learning had leveled, students had referred to what they had learned about developing 3D applications using Virtual World technology. We conclude that understanding of a complex software environment, merely by the formation of a community of learners, is unlikely to occur in a few weeks time.

Computing Education in a Hybrid World

RESEARCH QUESTIONS RQ6, RQ7

RQ6 Which subject-specific strategies were recommended in the past?

We have explored the effects of an educational approach that was common in Dutch undergraduate Multimedia education in 2008-2009. Students with little background in computing were exposed to advanced technology as authoring tools with minimal (if any) support. Lecturers assumed that learning communities would emerge, where students would share knowledge of specific, cutting-edge technology. Together, they would develop a professional approach to specific computing topics as 3D scripting.

RQ7 Which subject-specific strategies can we recommend?

We distinguish two possible strategies, depending on the course's learning objectives and the role of technology. If learning objectives do not include mastering advanced authoring tools because use of technology is only meant to illustrate concepts, students should be aware that they are not trained to develop working applications. In this variant, there should be a helpdesk where students can be informed about possible solutions to common problems, and be supported when facing more complex problems. Complex solutions can be faked, as long as the idea the student is elaborating is illustrated. Plenary presentations and assessments concern ideas rather than their implementation. In fact, this is what happened in the 2008 edition of the course, when one of the students, who already knew 3D technology, generously helped his colleagues. In the ideal elaboration of this variant, the use of technology is not assessed. Learning objectives do not include the competence to translate ideas in working prototypes, or, in today's terminology, minimum viable products. Students are aware they are not trained for development tasks, or professional use of digital technology.

If the learning objectives include mastering the tool because the course pursues the objective to translate innovative ideas in working digital products, the course should address implementation issues. In the case of the Virtual World course:

(1) students would have had access to resources about basic principles of 3D coding,

(2) the Active Worlds tool would have been introduced by stating the problems it addresses, their solutions and the downsides of these solutions, and

(3) basic notion of software quality would have been discussed, in order to safeguard further development of the product.

In this alternative, lectures discuss quality criteria of code in class. Some of the plenary presentations are devoted to common implementation problems and possible solutions. The use of technology is assessed. Students are aware that they are trained for professional use of authoring tools, and to produce (minimum viable) digital products.

RECOMMENDATIONS

We recommend lecturers of courses requiring the use of authoring tools to align the infrastructure of the course, its content and the assessment policy to the course's ambitions. We recommend them to state the role of technology explicitly.

FURTHER RESEARCH

We are aware of the important role online, informal communities of learners have played (and are playing) in the evolution of hybrid sub-disciplines of computing. It would be interesting to investigate if

- (1) Understanding of a complex software tool can be developed by a community of learners .
- (2) what such understanding consists of, and
- (3) is it reasonable to expect transfer of understanding to other scripting languages / authoring tools?

Part IV—HCI In A Hybrid Curriculum: Research In Action

RESEARCH QUESTION RQ7

RQ7 Which subject-specific strategies can we recommend?

In 2010/2011, we designed a course on Webculture for an audience, mainly consisting of students from the Department of Computer Science and students from the Department of Cultural Studies. The course had an important topic on design for the Web. The course addressed basic notions of Human Computer Interaction (HCI), and applied them to User Interface Design. Web technology was introduced by stating its history and its aims.

The students were academic distance learners. Some of them – those enrolled in a computing track – were freshmen, but the course was also open to senior students, enrolled in other programs. We wanted to involve all these students in our research activities, to allow them to witness research in action and discuss its outcomes.

But our intentions were too disruptive for our educational institution. Despite an encouraging pilot in a regular University, we were not able to test our instructional design. This section describes the ideas underlying the design of the online learning environment.

9 HCI in a Hybrid Curriculum: Research in Action⁷

ABSTRACT

Motivation – Tools that should support academic distance learning often support mainly distance teaching. In our vision, this is not compatible with the academic ambitions of university curricula. We propose an alternative approach.

Research approach – We share our observations from practice.

Findings/Design – We developed an environment for academic distance learning to allow adult students to experience research.

Research limitations/Implications – The setting, the Dutch University for Distance Learning, only allows action research.

Originality/Value – The researchers have experimented with online environments where they can perform research in the presence of their students.

Take away message – For distance learners too, academic education and research should go hand in hand. Online learning environments should be designed to support that.

KEYWORDS

Blended academic education; distance learning; Internet based learning environments; interaction design;

9.1 INTRODUCTION

The Open Universiteit, the Dutch Open University, is an institute for adult distance learning. The Computer Science Department developed an excellent blended curriculum that is considered as one of the European “best practices” (Sjoer, Veeningen, Jacobs, & de Jong, 2008). In the past few years, the focus switched from: how offer distance learners an up-to-date curriculum in Computer Science to: how to offer distance learners academic education.

⁷This work was originally published as: Benvenuti,L.,E.Rogier,G.C. van der Veer: " (Benvenuti, Rogier, & van der Veer, E-learning in a distance learning curriculum:a workplace approach, 2012)", Proceedings of the 2012 European Conference on Cognitive Ergonomics, Edinburgh, 29-31.08.2012. ©2012 ACM, NY, USA ISBN: 978-1-4503-1786-3 , doi: 10.1145/2448136.2448177

Computing Education in a Hybrid World

In this paper, we discuss our experiences in designing online courses that were meant to practice academic attitude. The approach is: take these students more seriously. Support them in their own exploration instead of giving them materials that have already been processed by somebody else.

This paper is on the development of student-centred educational resources. Merging action research (Kemmis & McTaggart, 1988) with education offers us the opportunity to witness the performance of the artefacts we design for distance education.

Sadly, an innovative approach is not always encouraged by the course development management that has to match educational goals with traditional business models and managerial constraints.

The result often is a compromise.

9.1.1 AN EXCELLENT CURRICULUM

The Open Universiteit's students are adult distance learners. At the Faculty of Computer Science, most of the students have full time jobs, many have families. They hardly have time to spend. Common motivations for these students to subscribe are: wish for promotion or need to certify one's experience (Menendez Blanco, van der Veer, Benvenuti, & Kirschner, 2012). The Open Universiteit allows these students to study at their own pace and at their own location, which is received as very successful. In the national ranking of suppliers of curricula for part-time higher education, the Open Universiteit has been on nr. 1 for the past few years (Centrum Hoger Onderwijs Informatie, 2012).

9.1.2 SUCCESS FACTORS

To investigate the success factors in the academic education of life long learners, E. Sjoer et al. (Sjoer, Veeningen, Jacobs, & de Jong, 2008) studied 5 best practices of blended curricula for lifelong learners. One of them was the Computer Science curriculum of the Open Universiteit. According to Sjoer, the critical succes factors are:

- The competence of the teachers
- Using collaborative activities, both online and face-to-face, to avoid lonely learners syndrome.
- The learning paradigm. The best paradigm is workplace learning, especially when authentic case material is used.

Sjoers conclusions at the educational level are that a vision for learning should be developed. Teachers should be educated in workplace learning,

supported in maintaining their didactical competence and guided on how to apply their knowledge to (e)learning processes.

9.1.3 CORPORATE GUIDELINES FOR THE E-LEARNING ENVIRONMENT

The Open Universiteit choose a proprietary system, Blackboard, as E-learning environment. Experiments with Moodle (Open Source) are conducted, but using Moodle as a live environment is viewed with suspicion. The policy on e-learning software is very cautious because the Electronic Learning Environment is a business-critical application. Today, the Open Universities workflow is completely based on Blackboard.

The Blackboard environment is used to support the educational process. Every course has a course site providing information on tutoring and exams. Communication between students is facilitated by forums. Virtual classrooms, that are commonly used, are accessible through the course site.

Course sites should be designed according to corporate guidelines to grant recognisability, both of the brand “Open Universiteit” and of the single Faculties.

Figure 9-1 shows one of the pages of the Computer Science course on Databases. The same page of most other courses of the Computer Science curriculum will differ only slightly from Figure 9-1: only the course name and code (encircled in Figure 9-1) will change.



Figure 9-1 Page in a standard course site

Computing Education in a Hybrid World

9.1.4 DISCUSSION

The Open University uses its Electronic Learning Environment to help the students to take courses efficiently and to support the lecturers in the delivery of course material. The workflow is moulded to fit to this approach. Here, we remark that efficiency in learning does not necessarily match with an academic attitude.

Most distance learners at the Computer Science Department are already working in ICT. ICT is evolving rapidly, hence their need for further education. Some students never were educated as computer scientists, but switched from other disciplines and learned on the job. They need certification of their expertise.

If the curriculum's aim is either refreshing knowledge or certification, a checklist-approach is appropriate: the course material will tell the students exactly what they have to know to pass the exam and will structure these items in a way that allows to process them rapidly. The focus on the teaching process fits these goals.

But the Open University aims at an academic curriculum. In that case the checklist approach is not enough. We focus on the essence of the academic practice: research and debate. We advocate a change of perspective from academic training (by the lecturer) to academic learning (by the student). Our aim is to design a rich learning environment for our students, an environment that supports exploration and peer "teaching" through forum discussion. Above all, we aim to design a distance research laboratory where the lecturers can perform their research in the presence of their students.

There is a lot of practice expertise among these students. Involving them structurally in their lecturers' research on how to optimally support distance learning can be inspiring.

9.2 A WORKPLACE ONLINE

Figure 9-2 shows a page of a course we designed on Webculture. The course, covering topics on design for the Web, is mandatory in the first year of the Computer Science curriculum but is also open to students from other Faculties. The course aims to improve the students' Web literacy and their participation in the online world. Also, it strives to sharpen awareness of the basic issues in design for the Web.

For this course, we designed an online workplace in Moodle, meant to stimulate exploration and exchange of opinions. The idea was to include written materials where needed, but to locate the course itself on the Web.

By keeping track of student's behavior and feedback, we would be able to validate our design choices.



Figure 9-2 Page of the Web-based course on Webculture

Unfortunately, the approach did not work in this environment. The Web-based version of the course was successfully tested in a regular University but was never released. A written version of this course will soon be published instead, accompanied by a course site in Blackboard (Figure 9-4). Some features of the Moodle version are implemented in this course site. In particular, we still can allow students to participate in our research program.

9.2.1 ACTUAL EXPERIMENT (BLACKBOARD COURSE SITE)

One of the areas investigated by the research group, is: design patterns for learning activities in a multimodal e-learning environment. Design patterns describe solutions to problems that occur frequently (Alexander, 1977) and the rationale for each solution, to support the making of motivated choices. We set up an experiment to investigate how different learning activities can be supported in multimodal distance e-learning environments.

Multimodal mini-courses of few minutes long were added to the course site as additional (optional) materials, explaining techniques or concepts that are taught in the course. We were interested in the students' appreciation for these additional resources.

Computing Education in a Hybrid World

Different modalities were implemented, including video courses, video with synchronised adjacent slides, text files, and demonstrations of websites with narration. Sometimes, several modalities are integrated, i.e. video, slides with voice over, presentation by a speaker while the slides are visible.



Figure 9-3 Mini course, added for research purposes

After watching the mini-courses, the students were asked to fill in questionnaires. This way, we investigated the following parameters in the learning material:

- Meaning: is it understandable?
- Attitude/emotion: is it fun?
- Attractivity: is it aesthetically pleasing?
- Engagement: Does it encourage you?

9.3 DESIGN ISSUES AND SOLUTIONS

To design an online workplace, suitable to our target audience, we formulated design recommendations for the course site, based on extensive literature research (Menendez Blanco, van der Veer, Benvenuti, & Kirschner, 2012).

The main goal of the workplace is: stimulate an academic attitude in the approach of our discipline. We summarize our strategy: (1) Put context and content both central in the design of the learning environment. (2) Encourage students to participate in the research program on free distance

HCI in a Hybrid Curriculum: Research in Action

learning. (3) Share results. (4) Practice what you preach and ask for feedback. (5) Apply new concepts while adapting to organisational constraints.

9.3.1 CONTEXT AND CONTENT BOTH CENTRAL

Following the principles that knowledge is most meaningful when rooted in a relevant, scaffolded context and that understanding is most relevant when rooted in personal experience (Hannafin & Land, 1997), we decided to support students in their exploration of the Web. We designed a structured online environment to return to in case the exploration would not be fruitful. This was translated in a content centred course site, where the table of content is always visible in the left column (Figure 9-2).

The course site is page based. Subsections that do not fit in one screen, are cut in pages that can be accessed sequentially. This is done to maintain the same visual design through the course, supporting the learning context.

Adult distance students have busy lives. We cannot assume that all our students are able to attend activities at the same time. Still, we occasionally want to encourage discussions, in the context of an assignment or as a tool to sharpen understanding of a particular issue. To support asynchronous communication in a shared educational context, we designed a right column for “tools” with access to a discussion forum (Figure 9-2).

9.3.2 ENCOURAGE STUDENT PARTICIPATION IN RESEARCH

Our aim is to gradually encourage students' participation in research. Webculture is mandatory to the first year, student involvement is mainly passive here. By filling the questionnaires, students can see how actual research is being done. They can experience how research topics are presented, how experiments are set up, how to make questionnaires and how responses on the experiments are acquired.

In the following years involvement in research can grow into a more active role. In other courses, we have asked students to explore certain topics themselves and to present the results to each other. These results are discussed by both the students and the lecturer. Elements that are particularly well done in a certain presentation are highlighted to allow others to learn from it.

Computing Education in a Hybrid World

9.3.3 SHARE RESULTS

Include recent publications of the lecturers' relevant research in the material and encourage discussion between students as well as discussion with the authors.

9.3.4 PRACTICE WHAT YOU PREACH AND ASK FOR FEEDBACK

The Moodle course was tested in a traditional university context. A welcome-message on the homepage stated the design philosophy and incited the students to provide us with comments about the design of the course site. The response was not massive (10%) but of excellent quality and led to improvements. Two students observed that they probably had not been able to find all the pages of the course; one remarked that even if he had done so, he was not aware of it. We added a progress indicator to the menu, showing for each section whether all the pages have been visited by the user or not (Figure 9-2). Its colour code is based on the traffic light metaphor.

9.3.5 APPLY NEW CONCEPTS WHILE ADAPTING TO ORGANISATIONAL CONSTRAINTS

While adapting to organisational constraints, be creative in optimally applying new opportunities and new understanding. In the Blackboard site, learning in context is supported by using one of the existing features (Learning Module) showing partial tables of content near the corporate buttons (see Figure 9-4).

Not all content can be integrated in this structure. Where integration is preferable but not possible, the second best solution is: open a new window. By closing that window, the student will return to the course site providing context information.

Computing Education in a Hybrid World

The course is performing well, students are triggered by the content they value as interesting, significant, stimulating. The additional resources are appreciated by the students, but active participation is lower than in fulltime education.

9.4.1 CONCLUSION

Our aim was to create an online resource supporting exploration and facilitating student participation in an online research lab. We partially succeeded.

Our aim was also: releasing an online resource suitable to test our ideas on design patterns for free distance learning. This part was less satisfactory. It would be easier to accomplish this goal if we were able to use Moodle. In that case, we would be able to adapt the tool to our ideas, design our own course sites and collect data from our students. This is not the case.

But even then, it is possible to apply new ideas, grown by new understanding, to the design of new course sites. Validating ideas is still possible, but more restrictedly. To do so with the course on Webculture, we had to step back, re-consider our design choices for the Moodle-site and translate them in Blackboard where possible.

ACKNOWLEDGMENTS

We thank our students for their feedback.

PART IV - Conclusions

In this Part, we describe a course on Webculture we designed for an audience of freshmen, enrolled in an academic program on Computer Science, and senior students, enrolled in an academic program Cultural Studies.

RESEARCH QUESTION RQ7

RQ7 Which subject-specific strategies can we recommend?

The course on Webculture illustrates an inquiry-based approach to computing targeting a hybrid audience. Students were freshmen, enrolled in a computing program, and students, enrolled in other programs as Cultural Studies. The inquiry-based approach was adopted to support distance learners in developing an academic attitude. Students were introduced to the essence of the academic practice, research and debate, by stimulating them to participate in the lecturer's research program. With the design of the course on Webculture, we show how participation of freshmen students can be achieved in a research program on Human Computer Interaction.

More in general, participating in activities related to knowledge enhancement supports understanding of the scope of knowledge. It helps students to better understand the evolution of the topics they are exploring, to approach literature and to keep up with future developments in their discipline. This applies both in academic and in professional setting, in distance learning as well as face-to-face, in full computing curricula and in hybrid curricula.

We were able to implement activities related to research in the course on Webculture. The course also aimed at engaging students in design activities, by allowing them to participate in the design of the resource's User Interface. This part was satisfactory in the test run. But in the end, the course was released on paper (Open Universiteit, 2012); no resources were allocated for further development of the course website. For these reasons, it was not possible to implement this ambition in the final course release.

RECOMMENDATIONS

- (1) Align context and content, encourage students' participation to research, share results, practice what you preach and ask for feedback.
- (2) Adapt to organizational constraints.

Computing Education in a Hybrid World

- (3) An inquiry-based approach as described in this Part requires that lecturers be actively engaged in knowledge enhancement.

FURTHER RESEARCH

Our current aim is to implement a blended course on digital design in an applied undergraduate program, and to implement features, designed by students during the course, in the course Web site.

It would be interesting to investigate if this approach results in better understanding of state-of-the-art guidelines for digital design, in a more active approach to literature and in more insight in strengths and weaknesses of the design methods lectured in class.

10 Overall Conclusions and Recommendations

This thesis is a reflection upon undergraduate computing education. Computing professions change over time. Our understanding of learning evolves. Educational programs are frequently updated. In the past 20+ years, we have witnessed major changes in this area, as the emergence of hybrid computing professions. What should designers of computing curricula always bear in mind, and what is important for designers of hybrid computing curricula in particular?

We started with philosophical considerations about computing and about hybrid computing professions (chapter 2, (Benvenuti, L. & van der Veer, G.C., 2009)). At the same time, we investigated the effects of education in a Dutch University of Applied Sciences, and designed educational units for the Dutch Open University.

We investigated, if different approaches to computing result in the instantiation of the same mental model for an abstraction describing a specific situation, and found differences (Part II).

We investigated an educational approach that was commonly adopted in Multimedia Education: the craftsmanship-based approach. Advanced software was considered as “tooling”. Students were offered pointers of tutorials, but no theoretical introduction was offered to them about computing topics as coding, or software development. The underlying assumption was: the students will form a community of practice and will learn from each other. We found indications for the community, less for the learning (Part III).

We designed a course for distance learners that aimed at increasing student participation in our research activities, and reflected upon the role of research in higher education. (Part IV)

To explain our findings, we took an historical perspective (Chapter 3). We found differences in the way computing disciplines approach one fundamental problem. The problem is: computing requires working with abstractions. Unlike mathematicians, who work with institutionalized abstract concept, computing professionals currently define the abstractions they work with. This raises the question, how to sustain claims concerning these abstractions. The first joint Task Force on the Core of Computing (Denning, et al., 1989) describes three approaches to this problem: the

Computing Education in a Hybrid World

formal, the scientific and the engineering approach. Computing, states the Task Force, is a unique blend of interaction between these three approaches. According to Tedre and Apiola (Tedre & Apiola, 2013), each of these approaches involves a specific view on the discipline of computing and respects specific epistemological values. We called these approaches *the three cultural styles of computing*, and referred to these three cultural styles to type hybrid computing curricula.

Finally, we reflected upon computing education in general and upon the mission of Dutch Universities of Applied Sciences (chapter 4). That mission is: providing Dutch industry with skilled workforce. We reflected upon the implications of this mission for undergraduate education in a rapidly evolving field as computing.

In this chapter, we will revisit all research questions. We will recall the conclusion of Part I (chapters 2,3 and 4). We will comment on the conclusions of Part II (chapters 5 and 6), Part III (Chapters 7 and 8) and Part IV (Chapter 9), while keeping in mind the framework we have sketched in chapter 3 to describe the discipline of computing and its relation to hybrid curricula. We will formulate recommendations concerning the design of undergraduate computing curricula including hybrid computing curricula.

Our main research question is:

Which lessons can be learned from past and present undergraduate computing education, which can be applied in the design of future undergraduate computing curricula and hybrid undergraduate computing curricula in particular?

We have decomposed the main research question in:

- RQ1 What are possible approaches to computing and computing education?*
- RQ2 What are the aims of undergraduate computing education?*
- RQ3 What is the purpose of undergraduate computing curriculum recommendations series, i.e. of (RQ3a) international curriculum recommendations series and (RQ3b) curriculum recommendations series for Dutch Universities of Applied Sciences?*

Overall Conclusions and Recommendations

- RQ4 *Do students, who were educated in different computing traditions, develop the same mental models for the abstract concepts they work with? I.e., are different approaches to computing interchangeable?*
- RQ5 *How do students in a hybrid curriculum experience a craftsmanship-based approach?*
- RQ6 *Which subject-specific strategies were recommended in the past?*
- RQ7 *Which subject-specific strategies can we recommend?*

10.1 RESEARCH QUESTION RQ1

- RQ1 *What are possible approaches to computing and computing education?*

10.1.1 PART I

In part I, we introduced three cultural styles of computing that were first described by Wegner (Wegner, Three computer traditions: Computer technology, computer mathematics and computer science, 1970), then by Denning et al. (Denning, et al., 1989) and recently by Tedre and Apiola (Tedre & Apiola, 2013). These cultural styles of computing fulfill different roles in the development of the discipline. All cultural styles are born from an attempt to cope with the same fundamental problem: how can claims, concerning abstraction, be sustained? The theoretical cultural style addresses the question in a formal way. It describes abstract structures in an unambiguous way and investigates their properties. The scientific cultural style addresses the question: do our (abstract) models match with the world they intend to describe? The engineering cultural style addresses the question, how to design and implement reliable computer systems, a process that requires exchanging information about these abstractions. According to Tedre and Apiola, cultural styles embody different epistemological values, in particular different approaches to research.

Our main research question concerns computing education. Do frameworks for computing curricula acknowledge this triple cultural approach to the discipline? ACM/ IEEE explicitly does. One of the aims of the (international) ACM/IEEE curriculum recommendations series is to support the development of one, unified, discipline of computing. Therefore, ACM / IEEE requires all graduates to be acquainted with “the core of computing”, a

Computing Education in a Hybrid World

selection of topics grounding all its sub disciplines and covering three cultural styles.

The European e-Competence Framework e-CF, and the HBO-ICT framework for computing education at the Dutch Universities of Applied Sciences, are less explicit in their vision of the discipline. The aim of the frameworks is primarily the description of professional roles. The approach is competence-based. We warned for a too strict translation of these descriptions in computing curricula. As for the Dutch Universities of Applied Sciences, we feared that applied curricula, strictly defined in terms of professional competences, without addressing reflection upon the discipline, could result in professionals overlooking other approaches to computing than the cultural style they were educated in.

We have called “hybrid” these programs, devoting a substantial part of their curriculum to computing, but less than 50%. The framework for the new hybrid domain in Dutch Universities of Applied Sciences (HBO-Creative Technologies) does not state its relation to computing explicitly. The framework seems to approach computing from a scientific point of view when assessing stakeholder requirements. In section 3.5.2, we argued that Front End Development is a hybrid computing curriculum. The program trains students for the development of marketable digital artifacts. It adds an engineering approach to computing to the scientific approach required by the framework HBO-Creative Technologies.

The three cultural styles address fundamental questions. They embody different epistemological values. An umbrella view, covering three epistemological points of view, sustains the development of a unified discipline of computing. Acquaintance with content from three cultural styles of computing is necessary to better understand the discipline and to collaborate with colleagues from other sub-disciplines. It is legitimate to ask, if a triple approach to research is also necessary for practicing professionals. This is a question that should be investigated.

Hybrid computing curricula devote less than 50% of their effort to computing. They cannot cover the core of computing. In section 3.6, we pleaded for the demarcation of the scope of hybrid professions.

10.1.2 PART III

In Part III, we followed the Virtual Worlds course at the Utrecht University of Applied Sciences, to better understand success factors of computing education in hybrid setting.

Overall Conclusions and Recommendations

The 2008 and 2009 editions of the Virtual World course, considered the authoring tool adopted to make digital artifacts (Active Worlds), merely as an instrument. The digital artifacts made by students were considered as prototypes, not meant for further development but to illustrate ideas. Quality criteria for code were not discussed, nor were implementation problems and solutions.

Retrospectively, we would not type the education offered in the Virtual Worlds course as “computing education”, because it did not address the question, how to handle abstraction. None of the following questions identifying the cultural styles of computing were addressed:

- how to describe abstract structures in an unambiguous way.
- Does an abstract model match with the world it describes?
- What does it take to design and implement reliable digital artifacts?

In our view, the students of the Virtual World course were not trained for professional use of the adopted tool, and were not trained to make reliable digital artifacts. They were trained to produce prototypes, rather than software products. To fulfill minimal quality criteria as maintainability or scalability, these prototypes ought to be re-implemented by professional software developers.

10.1.3 RECOMMENDATIONS

We recommend that all educators of computing topics understand the complex nature of computing. This applies to hybrid curricula and to undergraduate curricula at Dutch Universities of Applied Sciences in particular.

We recommend the Dutch HBO-I association, owner of the HBO-ICT framework, to stress the importance of “reflection upon (the evolution of) technology and upon the discipline of computing” in the next version of the HBO-ICT framework.

We recommend designers of hybrid curricula to refer to the three cultures of computing to type the curricula they design. It is inevitable for designers of hybrid curricula to make choices: which part of computing should be included, and why? Typing the computing related part of a curriculum by referring to its orientation gives insight in the related trade-offs.

We recommend educational programs, not addressing any of the fundamental questions of computing, to cultivate students’ awareness of the scope of their programs.

10.2 RESEARCH QUESTIONS RQ2, RQ3

RQ2 What are the aims of undergraduate computing education?

RQ3 What is the purpose of undergraduate computing curriculum recommendations series, i.e. of (RQ3a) international curriculum recommendations series and (RQ3b) curriculum recommendations series for Dutch Universities of Applied Sciences?

10.2.1 PART I

RQ2 What are the aims of undergraduate computing education?

In Part I, we compared different frameworks for undergraduate computing education. We have found different ideas about the aims of undergraduate computing education. One view is, that undergraduate computing education should ensure further development of the discipline. The other view is: computing education should train professionals able to apply state-of-the-art knowledge. Different stakeholders of the educational system will emphasize different aims: Academia will ask for intellectual development, Industry will ask skilled workforce, students will ask education offering them a sustainable view on the discipline.

The international ACM/IEEE curriculum recommendations series points towards both directions. In 1989, the ACM/IEEE Task Force on the Core of Computing stated that undergraduate computing education has three main stakeholders: its students, industry and the discipline itself. Students should gain access to the Master's level and to the labor market; they should be well equipped to keep up with future developments. Industry needs skilled manpower. According to ACM/IEEE, the development of the discipline of computing goes hand in hand with the definition of educational programs for professional figures.

Although the European e-Competence framework is not a curriculum framework, it is explicitly meant as a guide for European educational institutions. It emphasizes the output of education, in terms of competencies.

The Netherlands has a dual system. Academia focuses on research and development; it is responsible for the development of the discipline of computing. Universities of Applied Sciences, mainly prepare students to enter the labor market. Their computing programs focus on the application of state-of-the-art knowledge. Their graduates are not trained to develop the discipline but to contribute to the professionalization of the computing practice at hand.

Overall Conclusions and Recommendations

RQ3a What is the purpose of undergraduate computing curriculum recommendation series: international series.

We investigated the ACM/IEEE curriculum recommendations and the European e-Competence framework e-CF.

According to ACM/IEEE, the goal of education in general is to gain competence in a domain. The domain of computing is complex, because computing is a rapidly evolving intellectual discipline that can be approached from different points of view. With the curriculum recommendations series, ACM/IEEE aims to foster discipline oriented thinking. All computing professionals should be acquainted with a common core of computing, defined and updated by ACM/IEEE committees.

e-CF is not primarily a curriculum framework, but an industry standard. It aims at preventing shortage of qualified ICT manpower, by establishing a common language to express professional roles and competencies across Europe. Computing curricula are not standardized in this system.. e-CF emphasizes the output of education and pays less attention to other possible purposes of education.

RQ3b What is the purpose of undergraduate computing curriculum recommendation series: series for Dutch Universities of Applied Science?

The Dutch Universities of Applied Sciences (HBO) train professionals to apply knowledge that was developed in Academia. The HBO-ICT framework describes computing curricula for the Universities of Applied Sciences. The frameworks' principal aim is to support designers of curricula in the specification of programs for the education of computing practitioners, needed by the Dutch (and pan-European) labor market. This also applies to the new Dutch framework for hybrid curricula we have viewed, HBO-Creative Technologies.

The last edition of the HBO-ICT framework (HBO-ICT 2014) (a) trains computing professionals for the pan-European labor market, (b) having a common theoretical foundation, described in terms of state-of-the art technology (c) having extensive learning skills (d) able contribute to the professionalization of their profession.

We asked ourselves if definition of a "common theoretical foundation" should include the epistemological differences between cultural styles of computing, and concluded that this question still ought to be addressed.

The Dutch framework for curricula in the Creative Technologies (HBO-Creative Technology, hybrid) aims at boosting innovation by supporting the

Computing Education in a Hybrid World

Dutch economic top sector Creative Industries. It does not state its relation to computing explicitly.

10.2.2 PART III

RQ3b What is the purpose of undergraduate computing curriculum recommendation series: series for Dutch Universities of Applied Science

In Part III, we investigated the students' learning experience in a course on Virtual Worlds at the Utrecht University of Applied Sciences. We pointed at a possible ambiguity about the course's learning objectives. The curriculum on Digital Communication at the Utrecht University of Applied Science does not exist anymore. Its successor's name is Communication and Multimedia Design, one of the programs of today's domain of Creative Technologies. We will refer to the Virtual World course to illustrate inconsistencies in the ambitions of the domain HBO-Creative Technologies.

In the Virtual World course, the stated learning objectives concerned the concepts behind 3D Virtual Worlds, not implementation issues. In section 10.1.2, we have concluded that in our view, the course was not a course on computing because it did not address any of the fundamental questions of computing. The domain HBO-Creative Technologies gives room for this position towards computing. But the Virtual Worlds course might very well have looked as a course addressing computing topics to its students. Their future employers might have shared this perception.

Students were asked to use an advanced authoring tool to make a prototype implementing Virtual World concepts. Authoring tools support the implementation of proven, well-known features. Students who have innovative ideas need to expand the tools' boundaries. They will try to better understand the software they are working with, to add functionality. Their learning objectives for such an assignment are likely to be related to computing.

Designers of innovative applications can of course fake innovative features and deliver prototypes, which will be implemented by developers. But such a process supposes an articulated workflow. Small innovative companies, the companies targeted by the domain HBO-Creative Technologies, are more likely to work ad hoc. They are more likely to implement small working applications and further develop them in case of success. Employers in the Creative Industries are likely to expect basic development skills.

Overall Conclusions and Recommendations

10.2.3 RECOMMENDATIONS

e-CF describes the output of the educational system. Other stakeholders' requirements, as the students' need for a sustainable career perspective, or the discipline's long-term perspectives, seem in danger of being overlooked. European educational institutions offering computing programs should cope with this issue. This applies in particular to Dutch institutions offering applied computing programs.

We recommend (Dutch) national and international organizations to explore the question, of how differences in the research values of the three cultural styles of computing relate to the education of computing practitioners.

We recommend HBO-I to define a sustainable theoretical base for its curricula in collaboration with Academia. We recommend Dutch government to support the collaboration between institutions offering computing curricula in Academia and in the applied domain.

We recommend the domain HBO-Creative Technologies to fully clarify the professional roles its graduates will fulfill. Which programs will train designers of prototypes that will be implemented by developers, which ones to produce (minimum viable) innovative software products? The programs, which aim at training professionals to produce (minimum viable) products, should state their relation to computing.

We recommend designers of hybrid computing curricula to acknowledge that their programs are related to computing. We recommend them to take responsibility for the definition of new professions, and to participate in their development, in collaboration with related computing disciplines, Academia, the industry and the public authorities. We also recommend stating the scope of these new professions explicitly. This applies in particular for the programs of the Dutch domain of Creative Technologies.

We recommend designers of hybrid curricula to include the competence "understanding the limits of the acquired competences" among their final qualifications.

10.3 RESEARCH QUESTION RQ4

RQ4 Do students, who were educated in different computing traditions, develop the same mental models for the abstract concepts they work with? I.e., are different approaches to computing interchangeable?

Computing Education in a Hybrid World

10.3.1 PART II

RQ4 Do students, who were educated in different computing programs, develop the same mental models for the abstract concepts they work with? I.e., are different approaches to computing interchangeable?

We found indications for differences between the way students, enrolled in different computing curricula, conceptualize the abstract notion of “object”. The differences we found were not the differences we had expected.

Overall, we did not find awareness about the possible ambiguity of the abstract notion of “object”. In our opinion, this is an issue education should address.

10.3.1.1 The differences we had expected

In our preliminary study (chap. 2), we had described different ways to approach the computing practice: the formal approach to software requirements of Information Systems vs. the engineering approach of Software Engineering. We hypothesized different conceptualizations of abstract concepts by computing practitioners, educated in different programs.

We tested our hypothesis with the experiment described in chapter 5 and 6. We targeted senior students, enrolled in programs Business IT and Management (BIM) and Software Engineering (SE) at the Utrecht University of Applied Sciences. Senior students Information Engineering (IE) also were present at the time. They were included in the sample.

We had expected to find differences in problem solving preferences across these groups. We also had expected to find differences in mental models for the notion of “object”, and had expected that these differences would go hand in hand with differences in the computing practice. We had expected students BIM to conceptualize objects as sets, in analogy with the relational model. We had expected students SE to focus on entities, handled in software development. We had hypothesized that they would perceive instances, and would conceptualize them as bags of objects.

10.3.1.2 The differences we found

Within the limits set by the sample size, we found reliable indications for differences between the way students, enrolled in different curricula,

Overall Conclusions and Recommendations

conceptualize the abstract notion of “object”. We did not find significant differences in problem solving preferences across the groups.

We found indications for different mental models for the notion of object (sets or bags), although less reliably. SE students showed a preference for “set”, IE students for “bag”, the answers of BIM students were divided ad random between the options. This did not match our expectations. Within the same reliability range, we found indications for differences in the approach to abstractions related to software. 90% of the SE students provided implementation information about the system (as code, or a class diagram), against 70% of the IE students and 50% of the BIM students

10.3.1.3 A possible explanation

We had expected BIM students to conceptualize objects as sets, and SE students to conceptualize them as bags. But we found no evidence of such differences between BIM and SE students. We did find indications for these differences (set or bag) between groups of students. But these differences did not concern BIM students and SE students. They concerned SE and IE students.

A possible explanation refers to Tedre and Apiola’s framework of three cultural styles of computing (chapter 3). Dael (Dael, 2001) distinguishes two traditions in the development of educational programs for computing professionals in the Netherlands. He describes mighty battles between the factions in the first decades of the computing era. No consensus could be reached in the Netherlands on this issue until the 1990es. Dael traces the origin of computing programs in Dutch higher education back to the tradition that had started them.

One of the factions approached computing as an asset in business administration. The group was originally hosted by the Faculty of Economics and Business of the University of Amsterdam. In 2001, its heritage can be found both in Dutch Academia and the Universities of Applied Sciences. The other faction adopted a formal point of view. It was housed in the Mathematisch Centrum in Amsterdam. In 2001, its heritage can be found in Dutch Academia, not in Universities of Applied Sciences. A third faction (that Dael calls “the technicians”) did not participate at all in the debate about education, but formed a significant factor in Dutch industry.

It seems reasonable to assume that today’s Business IT and Management program in Dutch Universities of Applied Sciences has its origin in the business tradition (approaching computing from a scientific point of view),

Computing Education in a Hybrid World

and to track Software Engineering back to “the technicians” (professional figures requested by industry, who probably approached computing from an engineering point of view). But these are suppositions that should be investigated.

As for IE, the participants in our experiment had enrolled between 2009 and 2013. As we saw in chapter 0, the 2009 edition of the HBO-ICT framework emphasized specialization, and sustained the development of new computing programs. IE was one of these programs. It does not exist anymore today, but at the Utrecht University of Applied Sciences its know-how is maintained by the interest group Front End Development. In section 3.5.2, we argued that Front End Development’s approach to computing is partially engineering and partially scientific.

If we relate the results of the experiment in Part II to the programs’ cultural styles of origin to (instead of relating them to approaches to the computing practice), the overall picture changes. Let us assume that the cultural style of the faction, who once had initiated a program, still was a key factor to understand the effects of computing programs at the Dutch Universities of Applied Sciences in 2014. In that case, SE’s cultural style would have been predominantly engineering. BIM’s cultural style would have been predominantly scientific. Based on other arguments, we have stated that IE’s cultural style is likely to have been partially engineering and partially scientific.

If we put the programs in order, from “predominantly engineering” to “predominantly scientific”, we obtain this sequence: SE – IE – BIM. The results of our experiment at the Utrecht University of Applied Science reflect precisely this sliding scale. “The system” was described by providing implementation information by 90% of the students SE, 70% of the students IE and 50% of the students BIM. Within the limits of our scores’ reliability, these differences seem to be significant. The scientific cultural style of computing seems to go hand in hand with representing computer systems as black boxes, the engineering cultural style with representation as white boxes.

We explain the set/bag differences we have found by looking at the software layer the different educational programs focus at. IE focuses on designing and developing user interfaces. IE students mainly work with instances of objects; the “bag” was their preferred structure. SE programs focus more generally on “software”, which includes storage. As we remarked in section 6.7, the ‘Bookstore’ example is a typical case, used in courses on Databases. SE students probably referred to stored objects when describing

Overall Conclusions and Recommendations

“sets” of objects. BIM is not a design program, BIM students showed no preference for set of bags.

10.3.1.4 Hypothesis and further research

Our hypothesis concerning computing programs in Dutch Universities of Applied Sciences is that some of them (notably BIM, SE) still approach computing from different points of view. We suspect that these differences reflect the different traditions of the factions who, according to Dael (Dael, 2001), initiated them in the second half of the past century.

Our new hypothesis about mental representations of abstract concepts in computing is that students, educated in programs favoring different cultural styles of computing, seem to adopt different mental models of the same abstract concepts.

We found indications for such differences, but the samples we were able to collect were small and the interrater reliability was no more than fair. It would be interesting to repeat the experiment.

We have many open questions. We presume that the differences we have found result from different educational settings. They might also be pre-existent to the enrollment in different computing programs. Further research is needed to assess this.

Further research is also needed to investigate if the differences we presume are also present in other settings: in Dutch academic curricula, in foreign curricula implementing ACM/IEEE's core of computing, in computing practice.

How far is it possible to combine different abstract models during work operations? Combining models while discussing with colleagues might just be too complex for the human mind, even if the terminology is optimized.

All these issues require further research.

10.3.2 RECOMMENDATIONS

We recommend designers of computing curricula to aim at cultivating awareness for possible differences between conceptualizations of abstractions

10.4 RESEARCH QUESTION RQ5

RQ5 How do students in a hybrid curriculum experience a craftsmanship-based approach?

Computing Education in a Hybrid World

10.4.1 PART III

RQ5 How do students in a hybrid curriculum experience a craftsmanship-based approach?

In 2008-2009, a craftsmanship-based approach was common in Dutch undergraduate Multimedia education. Students with little background in computing were exposed to advanced technology as authoring tools with minimal (if any) support. Lecturers assumed that learning communities would emerge, where students would share knowledge of specific, cutting-edge technology. Together, they would develop a professional approach to specific computing topics as 3D scripting.

Although we do not exclude that this strategy could work in limited settings, we warn against too enthusiastic expectations. In 2009, we investigated the students' learning experience in a course on Virtual Worlds, which seemed to have enabled the emergence of such a community. We found indeed that during an assignment in a shared virtual world, the sense of community among the students had increased significantly. But the students also reported that their learning had not increased during the assignment in the virtual world.

We think that when the students had stated that their learning had leveled, they had referred to what they had learned about developing 3D applications using Virtual World technology. Sadly, the course was drastically redesigned in 2010. It was not possible to test our hypothesis. We have open questions about this craftsmanship-based approach to education involving authoring tools. It would be interesting to investigate if

- (1) Understanding of a complex software tool can be developed by a community of learners .
- (2) what such understanding consists of, and
- (3) is it reasonable to expect transfer of understanding to other scripting languages / authoring tools?

10.4.2 RECOMMENDATIONS

We warn against too enthusiastic expectations about the role of communities as autonomous supports for learning activities concerning complex software.

10.5 RESEARCH QUESTIONS RQ6, RQ7

RQ6 Which subject-specific strategies were recommended in the past?

Overall Conclusions and Recommendations

RQ7 Which subject-specific strategies can we recommend?

10.5.1 PART I

RQ6 Which subject-specific strategies were recommended in the past?

The first Task Force on the Core of Computing recommended an inquiry-based approach to education. Rather than lectures presenting answers, the Task Force recommended acquaintance with the computing literature and with the related research methods. But according to Tedre and Apiola, the three cultural styles have fundamentally different approaches to research. Up-to-date knowledge of one of the sub disciplines of computing and acquaintance with the related research methods is more likely to point towards a specialist career path than towards overview of the discipline. We doubt that an inquiry-based approach in the undergraduate curriculum can be combined both with the Task Forces' aim to foster discipline oriented thinking and its goal to train students to access the labor market.

The Task Force also recommended using differences between programming languages (or programming paradigms) as a vehicle to discuss differences between approaches to computing.

Tedre and Apiola recommend aligning learning objectives with the cultural style because a mix would not result in successful educational interventions.

RQ7 Which subject-specific strategies can we recommend?

At course level, we endorse Tedre and Apiola's recommendation to align learning objectives with the cultural style, but we also recommend to cultivate awareness for differences in cultural style in computing curricula.

The fundamental question "how can claims, concerning abstraction, be sustained?" also applies to digital technology itself, and can be approached in different ways: matching a theoretical, engineering or scientific approach. In line with this, we recommend all computing curricula, and the HBO-ICT curricula in particular, to address the problems behind the development of technologies, rather than focusing on knowledge of state-of-the-art technology. Approaching technology as an illustration of more general principles will support the graduates' future understanding of the discipline.

10.5.2 PART II

RQ7 Which subject-specific strategies can we recommend?

Computing Education in a Hybrid World

We recommend all computing curricula to explicitly address the question, “how can claims, concerning abstraction, be sustained?”

We recommend educators and practitioners to establish and use a refined terminology for the notion of “object”, in order to improve recognition of different mental models.

We recommend all computing curricula to explicitly cultivate awareness for possible different conceptualizations of abstract concepts.

10.5.3 PART III

RQ6 Which subject-specific strategies were recommended in the past?

We have explored the effects of an educational approach that was common in Dutch undergraduate Multimedia education in 2008-2009. Students with little background in computing were exposed to advanced technology as authoring tools with minimal (if any) support. Lecturers assumed that learning communities would emerge, where students would share knowledge of specific, cutting-edge technology. Together, they would develop a professional approach to specific computing topics as 3D scripting.

RQ7 Which subject-specific strategies can we recommend?

To support each other in their learning process, the members of a learning community should agree with each other (and with their the lecturers) upon the objectives that are pursued. The course objectives should be aligned with the challenges, students face while using technology.

If the use of advanced technology is meant to illustrate concepts, instead of making working products, the students should be enabled to devote their attention to design challenges. They should be supported when facing challenges concerning the implementation of their prototypes. In such courses, solutions to implementation problems (scripting) will not be assessed. Further development of the prototypes, made by students, is not expected. In section 10.1.2, we have argued, that this course should not be typed as “computing education”.

If the course’s learning objectives are: translating innovative ideas in working products, or minimum viable products, solutions to implementation problems should be assessed. The course should introduce guidelines for coding, as guidelines aiming at improve the lifetime expectancy of software, or its flexibility. The reasons for these guidelines should be stated. For such courses, more generally for courses in in hybrid setting pursuing the objective to translate ideas in working digital products, we recommend approaching computing from a mixed engineering/scientific

Overall Conclusions and Recommendations

point of view. We recommend to align educational design to the engineering cultural style for software development, and to the scientific cultural style when discussing software validation. Educational resources and assessment policy should be aligned with the related cultural style. Students should be aware that they are trained to make (minimum viable) software products. The technology itself should be introduced by stating its cultural approach to computing, which enables a discussion about its scope. In all cases, we recommend to discuss the scope of the students' future expertise in class.

10.5.4 PART IV

RQ7 Which subject-specific strategies can we recommend?

In the context of this dissertation, chapter 0 provides an example of an inquiry-based course targeting a hybrid audience in academic setting. The course treated basic notions of HCI, applied to User Interface design. It targeted an academic audience, and strived at engaging students in research activities. Computing was approached from a point of view compatible with the content: a discussion of general notions of HCI was combined with a scientific approach to research.

Part of the students consisted of freshmen, enrolled in a computing program. These students were better acquainted with an engineering approach to computing than with the scientific approach. The alignment between content and cultural style illustrates how scientific knowledge is generated. It paves the way for a discussion of its values, its possible shortcomings and the comparison with other kinds of knowledge.

The course illustrates that alignment of course content and cultural style can be achieved in many circumstances. There are trade-offs, though. As we argued in section 3.4.1, an inquiry-based approach is likely to direct students' understanding of the discipline towards values, consistent with one of the three traditions. In this case, towards the scientific tradition's values (putting accuracy above utility) rather than the engineering tradition's values (putting utility above accuracy). Lecturers should be aware of the issue, and address it on request.

This strategy requests the lecturer(s) to be actively engaged in research in the topic they are lecturing.

The course, described in chapter 0, introduced Web technology by describing its history and its principles. Today, we would also point at two main aspects of W3C 'standards'. The - formal - definition of languages as HTML describes (in an unambiguous way) how browsers should work. W3C guidelines for correct semantic use of tags embody an engineering

Computing Education in a Hybrid World

approach. Applying those guidelines will enhance the quality of Websites and increase their lifetime expectancy. Those guidelines are not unambiguous; lively discussions do occur among Web developers upon how to apply them. Language definitions and W3C guidelines both support interoperability of software. They also have different aims and respect different values. Stating this would allow us to address the problems behind the development of Web technology, as recommended in section 10.5.1, and to point at differences between cultural styles.

10.5.5 RECOMMENDATIONS

The combination of the first ACM/IEEE Task Forces' aim to foster discipline oriented thinking, the curricular goal to train students for entering the labor market, and the recommended instructional strategy (inquiry-based) ought to be reconsidered.

We endorse the first ACM/IEEE Task Force's recommendation to use programming languages, programming paradigms or technology, as a vehicle to discuss differences between approaches to computing. In line with this, we recommend HBO-ICT to address the problems behind the development of technologies rather than focusing on knowledge of state-of-the-art technology. Approaching technology as an illustration of more general principles, including its cultural approach to computing will support the graduates' future understanding of the discipline.

As for hybrid computing curricula, we recommend that lecturers of computing topics (1) should have a CS degree or equivalent, (2) discuss aims and boundaries of hybrid programs with their students and (3) refer to technology as a vehicle for that discussion.

10.6 WHICH LESSONS CAN WE LEARN THAT CAN BE APPLIED?

Our main research question was:

Which lessons can be learned from past and present undergraduate computing education, which can be applied in the design of future undergraduate computing curricula and hybrid undergraduate computing curricula in particular?

Which lessons can be learned? In a few words: not to underestimate the consequences of computing related education. Abstraction is a key concept in computing. Leaving the understanding of abstractions to history, the

Overall Conclusions and Recommendations

market or emerging professional communities seems not to result in compatible abstractions. Neglecting this aspect of computing will result in a professional Tower of Babel. In a world, which is rapidly embracing the “computing for all” credo, it would be interesting to agree upon what we are talking about.

We have drawn attention towards factors that can augment the probability of a divergent scenario. Some of them directly relate to the way computing is taught. Their impact can be softened with training and certification programs for lecturers. But other factors are more difficult to deal with, because they involve many actors. We refer to the definition of professional roles, or the design of educational systems.

Two international professional organizations, ACM and IEEE, have identified this problem decades ago. They started a program that aims at supporting the development of one, unified discipline. This program has defined a list of topics that every graduate in computing should be acquainted with: the core of computing. To ACM / IEEE we would say: to foster discipline oriented thinking, understanding of the differences between possible approaches to research might be more important than the content. We called these possible approaches: the three cultural styles of computing. We stress the importance of discussing differences between cultural styles of computing more explicitly, because that discussion will foster awareness of computing’s complex nature, both of scholars and professionals.

We are concerned about the education of computing practitioners. Besides being equipped with a Body of Knowledge and Skills, practitioners should be able to perform (applied) research. But acquaintance with methods for applied research, related to a specific professional situation, is likely to point towards a specialist career path rather than towards overview of the discipline. Is it still realistic to aim at fostering discipline oriented thinking in the undergraduate computing curricula? If it is not, what does it mean to be a “computing professional”? These questions ought to be addressed.

Academic and applied education are separated in the Netherlands. A too strict separation seems not fruitful to us in the domain of computing. In today’s world, preparing students to find a job does not necessarily mean preparing them to keep up with their profession, or to design the future of their profession. We recommend Dutch Universities of Applied Sciences and Dutch Universities offering computing programs, to enhance their cooperation, in order to define the theoretical base of applied computing

Computing Education in a Hybrid World

curricula. We recommend the Dutch public authorities to sustain that cooperation.

As for hybrid computing curricula: it is inevitable to make choices while designing a hybrid curriculum. We recommend referring to the cultural styles of computing to type hybrid curricula. We also saw that defining and tuning undergraduate computing curricula is not only a matter of training manpower requested by industry. We saw that the field of computing was shaped by a joint effort of industry, Academia and governmental institutions, and urge designers of hybrid curricula to take responsibility for the definition of new disciplines. Trade-offs, involved by curricular choices define the boundaries of new professional roles. They should be discussed explicitly in broad communities including: related computing disciplines in Academia, the industry and the public authorities

This thesis concerns undergraduate computing education, in all its variants. Lecturing technology without referring to its philosophical background, lecturing computing without addressing the complexity of handling abstraction, would mean not taking students seriously enough to invite them to join the conversation about their own professional future. We have warned against a sloppy approach to the discipline.

REFERENCES

- ACM / AIS. (2010). *Curriculum Guidelines for the Undergraduate Degree Programs in Information Systems*. ACM.
- ACM / IEEE. (2004). *SE 2004: Software Engineering 2004, curriculum Guidelines for Undergraduate Degree Programs in Software Engineeri*. ACM / IEEE.
- ACM / IEEE. (2005). *Computing Curricula 2005, the Overview Report*. ACM/IEEE. Retrieved 6 10, 2016, from <https://www.acm.org/education/curricula-recommendations>
- ACM / IEEE. (2008). *CS 2008: Computer Science Curriculum 2008: An interim revision of CS2001*. ACM / IEEE.
- ACM / IEEE. (2013). *CS 2013: Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM/IEEE.
- ACM / IEEE. (2017). *IT 2017: Curriculum Guidelines for Undergraduate Degree Programs in Information Technology*. ACM/IEEE.
- ACM/IEEE. (2014). *SE 2014: curriculum guidelines for undergraduate degree programs in software engineering*. . ACM/IEEE.
- ACM/IEEE Task Force on Computing Curricula. (2014). *Software Engineering 2014, curriculum guidelines for undergraduate degree programs in SE (draft)*. Retrieved 06 20, 2015, from <https://www.acm.org/education/curricula-recommendations>
- Adobe Systems. (n.d.). *Acrobat Connect*. Retrieved may 28, 2008, from <http://www.adobe.com/products/acrobatconnectpro/>
- Alexander, C. (1977). *A pattern language: towns, building, constructions*. Oxford: Oxford University press.
- Antonacci, D., & Modaress, N. (2008). Envisioning the Educational Possibilities of User-Created Virtual Worlds. *AACE Journal*, 16(2), 115-126.
- Baddeley, A., Eysenck, M., & Anderson, M. (2009). *Memory*. Psychology Press.
- Bakker, G., Meulenber, F., & de Rode, J. (2003). Truth and Credibility as a Double Ambition: Reconstructions of the Built Past, Experiences and Dilemmas. *The Journal of Visualisation and Computr Animation*, 14, 159-167.
- Benvenuti, L., & van der Veer, G. (2011). Practice what you preach: experiences with teaching 3D concepts in a virtual world. In S. H.-J. (ed.), *Virtual Immersive and 3D Learning Spaces: Emerging Technologies and Trends* (p. Ch. 3). IGI-global.
- Benvenuti, L., & van der Veer, G. (2014). The Object Relational impedance mismatch from a cognitive point of view. In B. du Boulay, & J. Good (Ed.), *Psychology of Programming 2014*.
- Benvenuti, L., & van der Veer, G. (2014). The Object Relational impedance mismatch from a cognitive point of view. In B. du Boulay, & J. Good (Ed.), *Psychology of Programming 2014*.
- Benvenuti, L., & van der Veer, G.C. (2009). Multimedia design kan je leren. Kunnen we het ook doceren? In D. L. F.J. Verbeek (Ed.), *Change! - Proiceedings CHI-NL 2009* (pp. 39-41). Leiden, NL: CHI-NL.
- Benvenuti, L., Barendsen, E., van der Veer, G. C., & Versendaal, J. (2018). Understanding computing in a hybrid world, on the undergraduate curriculum Front End Development. *SIGCSE2018 / CS for All*. Baltimore: ACM.
- Benvenuti, L., Hennipman, E., Oppelaar, E. R., van der Veer, G. C., Cruijsberg, R., & Bakker, G. (2010). Experiencing and learning with 3D virtual worlds. In J. M. Spector, D. Ifenthaler, P. Isaias, & D. G. Kinshuk, *Learning and instruction in the digital age* (p. CH 12). NewYork, NY, USA: Springer Science+Business Media, inc.

Computing Education in a Hybrid World

- Benvenuti, L., Hennipman, E., Oppelaar, E., van der Veer, G., Cruisberg, R. '., & Bakker, G. (2008). Experiencing Education with 3D virtual worlds. In D. Kinshuk, D. Sampson, J. Spector, P. Isaías, & D. Ifenthaler (Ed.), *Proceedings of the IADIS International Conference on Cognition and Exploratory Learning in the Digital Age* (pp. 295-300). Freiburg, Germany: IADIS.
- Benvenuti, L., Louwe Kooijmans, C.F., Versendaal, J., & van der Veer, G.C. (2015). Representations of abstract concepts, differences across computing disciplines. *Frontiers in Education 2015*. El Paso, TX: IEEE.
- Benvenuti, L., Rogier, E., & van der Veer, G. (2012). E-learning in a distance learning curriculum: a workplace approach. *Proceedings of the 2012 conference on Cognitive Ergonomics*. Edinburgh.
- Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *The Unified Modeling Language, User Guide* (2nd ed.).
- Bruce, K., Cupper, R., & Drysdale, R. (2010). A history of the Liberal Arts and Computer Science consortium and its model curricula. *ACM Transaction on Computing Education*, 10(1), 3.
- Bryman, A. (2012). *Social Research methods* (4th ed.). New York: Oxford University press.
- Cañas, J. &. (1998). The role of working memory in measuring mental models. *Ninth European Conference on Cognitive Ergonomics*. EACE.
- Cassel, L. (2007). Understanding the entirety of modern Informatics. *Innovations in teaching and learning in computer science*, 6(3), 3-11.
- CEN. (n.d.). *ICT Profiles*. Retrieved March 29, 2018, from European e-Competence Framework: <http://www.ecompetences.eu/ict-professional-profiles/>
- Centrum Hoger Onderwijs Informatie. (2012). *Keuzegids Deeltijd & Duaal 2009, 2010, 2011, 2012*. Leiden, the Netherlands.
- Chan, H., Wei, K., & Siau, K. (1993). User-database interface: the effect of abstraction levels on query performance. *MIS quarterly*, 17(4), 441-464.
- Cobol. (2014, 05 25). Retrieved from Wikipedia: <http://en.wikipedia.org/wiki/COBOL>
- Comité de Suivre de la Licence. (2010, 12 29). *Recommendations Année 2007-2008*. Retrieved from <http://www.enseignementsup-recherche.gouv.fr/cid21521/remise-du-rapport-dizambourg-a-valerie-pecesse.htm>
- Commissie Accreditatie Hoger Onderwijs. (2011). *Prikkelen, presteren profileren*. the Hague, NL.
- Commission des Titres d'Ingénieur. (n.d.). *Higher Education in France*. Retrieved 12 28, 2010, from <http://www.cti-commission.fr/The-French-higher-education-system>
- Commission des Titres d'Ingénieur. (n.d.). *Les Écoles françaises d'ingénieurs (the French engineering schools)*. Retrieved 12 28, 2010, from <http://www.cti-commission.fr/IMG/pdf/GrandesEcoles.pdf>
- Conklin, M. (2007). *101 Uses of Second Life in the College Classroom*. Retrieved may 28, 2008, from <http://facstaff.elon.edu/mconklin/pubs/glshandout.pdf>
- Cortesi, A., & Nardelli, E. (2007). Towards an European Certification of Computer Science Curricula. *Innovations in teaching and learning in computer science*, 6(3), 79-86.
- Cowling, A. (2006). *A system model for the field of Informatics*. Retrieved 12 24, 2020, from http://www.ics.heacademy.ac.uk/education_europe/programme.htm
- Cross, J., O'Driscoll, T., & Trondsen, E. (2007). Another Life: Virtual Worlds as Tools for Learning. *eLearn 2007*, 3(2).
- Dael, R. (2001). *Iets met computers: over de beroepsvorming van de informaticus*. Delft: Eburon.
- de Haan, L., & Koppelaars, T. (2007). *Applied mathematics for database professionals*. (Apress, Ed.)
- Dede, C. (1995). The Evolution of Constructivist Learning Environments: Immersion in Distributed, Virtual Worl. *Educational Technology*, 35(5), 46-52.

References

- Denning, P. (2007, July). Computing is a Natural Science. *Communications of the ACM*, 50(7).
- Denning, P., Comer, D., Gries, D., Mulder, M., Tucker, A., Turner, A., & Young, P. (1989). Computing as a discipline. *Communications of the ACM*, 32(1), pp. 9-23.
- Denning, P., Feigenbaum, E., Gilmore, P., Hearn, A., Ritchie, R., & Traub, J. (1981). A discipline in crisis. *Communications of the ACM*, 24(6), 370-374.
- Detienne, F. (1997). Assessing the cognitive consequences of the object-oriented approach: A survey of empirical research on object-oriented design by individuals and teams. *Interacting with computers*, 9(1), 47-72.
- Dialogic & Matchcare. (2016, 3). *Digitaal Vakmanschap, van de ICT arbeidsmarkt naar de arbeidsmarkt voor ICT'ers*. Retrieved 11 02, 2016, from <https://www.nederlandict.nl/news/tekort-aan-developers-neemt-toe/>
- Dickey, M. (2005). Three-dimensional virtual worlds and distance learning: two case studies of Active Worlds as a medium for distance education. *British Journal of Educational Technology*, 36(3), 439-451.
- Dijkstra, E. (1973, may 23). *Programming as a discipline of mathematical nature*. Retrieved september 28, 2015, from <http://www.cs.utexas.edu/users/EWD/transcriptions/EWD03xx/EWD361.html>
- Dijkstra, E. (1988). *On the cruelty of really teaching computing science*. Retrieved 4 16, 2009, from <http://www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1036.html>
- Domein Creative Technologies. (2014). *Bachelor of Creative Technologies- beroeps en competentieprofiel domein Creative Technologies*. Vereniging van Hogescholen.
- du Boulay, B., O'Shea, T., & Monk, J. (1989). The black box inside the glass box; presenting computing concepts to novices. *International journal of man-machine study*, 14, 237-249.
- Educause Learning Initiative. (2006, june). *7 things you should know about Virtual Worlds*. Retrieved may 28, 2008, from <http://www.educause.edu/ir/library/pdf/EL17015.pdf>
- EduTech Wiki. (n.d.). *Constructivism*. (U. o. Geneva, Producer) Retrieved december 12, 2008, from EduTech Wiki of the University of Geneva: <http://edutechwiki.unige.ch/en/Constructivism>
- EHEA. (n.d.). Retrieved 12 29, 2010, from Bologna Process: <http://www.ehea.info/>
- Eliëns, A., Feldberg, F., Konijn, F., & Compter, F. (2007). VU @ Second Life: Creating a (Virtual) Community of Learners. *proceedings EUROMEDIA*. Delft, the Netherlands.
- European Commission. (2005, May). *Descriptors defining levels in the European Qualifications Framework (EQF)*. Retrieved March 29, 2018, from Learning opportunities and Qualifications in Europe: <https://ec.europa.eu/ploteus/en/content/descriptors-page>
- European Committee for Standardisation (2). (n.d.). *eCF founding principles*. Retrieved 6 22, 2016, from European eCompetence Framework: <http://www.ecompetences.eu/e-cf-founding-principles/>
- European Committee for Standardisation. (2014). *Context*. Retrieved 6 22, 2016, from European e-Competence framework: <http://www.ecompetences.eu/context/>
- European Committee for Standardization. (2014). *European e-Competence Framework v 3.0*. European Committee for Standardization. Retrieved 06 22, 2016, from European e Competence Framework v 3.0: <http://www.ecompetences.eu/>
- Food and Drug Administration. (2002, 01 11). *General Principles of Software validation; Final Guidance for Industry and FDA Staf*. Retrieved 09 18, 2010, from

Computing Education in a Hybrid World

- <http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm085371.pdf>
- Formal Sciences*. (n.d.). Retrieved 03 18, 2011, from Wikipedia:
http://en.wikipedia.org/wiki/Formal_sciences
- Freelon, D. (2013). ReCal OIR: Ordinal, interval, and ratio intercoder reliability as a web service. *International Journal of Internet Science*, 8(1), 10-16.
- Gaming Today. (2007). *Scientists Call WoW Corrupted Blood Epidemic a Disease Mode*. Retrieved may 28, 2008, from <http://news.filefront.com/scientists-call-wow-corrupted-blood-epidemic-a-diseasemodel/>
- Genova, G. (2010, July). Is Computer Science truly Scientific? *Communications of the ACM*, 53(7).
- Hannafin, M., & Land, S. (1997). The foundations and assumptions of technology-enhanced student-centered learning environments. *Instructional Science*, 25, 167-202.
- Hayes, E. (2006). Situated Learning in Virtual Worlds: The Learning Ecology of Second Life. *47th annual Adult Education Research conference*. Minneapolis, US.
- HBO-Raad. (2009). *Kwaliteit als Opdracht*. The Hague, NL: HBO-Raad.
- Hilbert. (n.d.). Retrieved 11 2010, 2010, from Wikipedia: <http://en.wikipedia.org/wiki/Hilbert>
- Hogeschool Utrecht. (2012). *Studiegids Bacheloropleidingen institute of ICT 2012-2013*. Utrecht, the Netherlands.
- Hogeschool Utrecht. (2013). *Studiegids Bacheloropleidingen Business IT & Management voltijd 2013-2014*. Utrecht, the Netherlands: Hogeschool utrecht.
- Hogeschool Utrecht. (2016). *EXIN en HU koplopers toepassing e-CF*. Retrieved 3 22, 2018, from Hogeschool utrecht: <https://www.hu.nl/overdehu/nieuws/EXIN-en-HU-koplopers-toepassing-e-CF>
- Hubwieser, P. (2013). The darmstadt model, a first step towards a research framework for computer science education in schools. *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 1-14). Berlin, Heidelberg: Springer.
- Hubwieser, P., Armoni, M., Brinda, T., Dagiene, V., Diethelm, I., Giannakos, M. N., . . . Schubert, S. (2011). Computer science/informatics in secondary education. In ACM (Ed.), *In Proceedings of the 16th annual conference reports on Innovation and technology in computer science education-working group reports*, (pp. pp. 19-38).
- IEEE, A. /. (2013). *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM/ IEEE.
- Informatics Europe. (n.d.). Retrieved 12 28, 2010, from Informatics Europe:
<http://www.informatics-europe.org/about.php>
- Ireland, C. (2011). Exploring the Essence of an Object-Relational Impedance Mismatch-A novel technique based on Equivalence in the context of a Framework. *The Third International Conference on Advances in Databases, Knowledge, and Data Applications*, (pp. 65-70).
- Johnson-Laird, P. (1989). Mental Models. In M. Posner, *The foundations of Cognitive Science*. Cambridge, MA, USA: MIT Press.
- Johnson-Laird, P., & Byrne, R. (n.d.). *Mental models, a gentle introduction*. Retrieved 02 8, 2014, from Mental models website: <http://mentalmodelsblog.wordpress.com/mental-models-a-gentle-introduction/>
- Joint Computer Conferences*. (n.d.). Retrieved 3 18, 2016, from https://en.wikipedia.org/wiki/Joint_Computer_Conference
- Jonassen, D., & Roher-Murphy, L. (1999). Activity Theory as a Framework for Designing Constructivist Learning Environments. *Educational Technology Research and Development*, 47(1), 61-79.

References

- Kamel Boulos, M., Hetherington, L., & Wheeler, S. (2007). Second Life: an Overview of the Potential of 3-D Virtual Worlds in Medical and Health Education. *Health information and Libraries Journal*, 24(4), 223-245.
- Kemmis, S., & McTaggart, R. (1988). *The action research planner*, (3rd ed). Victoria, Australia: Deakin University press.
- Kirriemuir, J. (2008, March). *Measuring the Impact of Second Life for Educational Purposes*. Retrieved may 28, 2008, from Virtual World Watch: <http://virtualworldwatch.net/snapshots/measuring-the-impactof-sl-in-educationmarch-2008/>
- Lakoff, G. &. (2000). *Where mathematics comes from: How the embodied mind brings mathematics into being*. (B. Books, Ed.)
- Lamont, I. (2007). *Harvard's Virtual Education Experiment in Second Life*. Retrieved may 28, 2008, from <http://blogs.computerworld.com/node/5553>
- Langley, G., & Sheppard, H. (1985). The visual analogue scale: its use in pain measurement. *Rheumatology International*, 5(4), 145-148.
- Larman, C. (2005). *Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development* (3rd ed.). India: Pearson Education.
- Larsen, V., & Lubbe, M. (2008). *Quick Scan jong talent en de Wetenschap*. VSNU.
- Lewis, C., Jackson, M., & Waite, W. (2010, May). Student and Faculty Attitudes and Beliefs About Computer Science. *Communications of the ACM*, 53(5), pp. 77-85.
- Liberal Arts and Computer Science Consortium. (2007). A 2007 model curriculum for a Liberal Arts degree in Computer Science. *Journal on Educational Resources in Computing*, 7(2), 2.
- Linden Lab. (2003). Retrieved may 28, 2008, from Second Life: <https://www.lindenlab.com/>
- Livingstone, D., & Kemp, J. (2006). Massively Multi-Learner: Recent Advances in 3D Social Environments. *Computing and Information Systems Journal*, 10(2).
- Loftus, T. (2005, 2 25). *Virtual World taches Real World skills*. Retrieved may 26, 2008, from NBC News: <http://www.msnbc.msn.com/id/7012645/>
- Lolli, G. (2006). La questione dei fondamenti fra matematica e filosofia. (S. Albeverio, & F. Minazzi, Eds.) *Note di matematica, storia, cultura*, 14-15, 17-35.
- Magnusson, S., Krajcik, J., & Borko, H. (1999). Nature, sources, and development of pedagogical content knowledge for science teaching. In *Examining pedagogical content knowledge* (pp. 95-132). Dordrecht, the Netherlands: Springer.
- Martinez, L. M., Martinez, P., & Warkentin, G. (2007). A First Experience on Implementing a Lecture on Second Life. *Second Life Education Workshop*, (pp. 52-55). Chicago, USA.
- Mason, H. (2007). Experiential Education in Second Life. *Second Life Educational Eorkshop 2007*, (pp. 14-18). Chicago, US.
- Mayer, R. (1989). The psychology of how novices learn computer programming. In E. Soloway, & J. Sphorer, *Studying the novice programmer*. Hillsdale, NJ: Lawrence Erlbaum.
- Menendez Blanco, M., van der Veer, G., Benvenuti, L., & Kirschner, P. (2012). Design Guidelines for Self-assessment Support for Adult Academic Distance Learning. In S. Hai-Jew (Ed.), *Constructing self-discovery Learning Spaces online: Scaffolding and Decision making Technologies* (pp. 169-198). IGI-global.
- Ministero dell'Istruzione, dell'Universita' e della Ricerca. (2000, 10 19). Decreto Ministeriale 4 agosto 2000: Determinazione delle classi delle lauree universitarie, . *Gazzetta Ufficiale*, 245. Retrieved from MIUR - Atti ministeriali: Decreto Ministeriale 4 agosto 2000: Determinazione delle classi delle lauree universitarie, Gazzetta Ufficiale 19 ottobre 2000 n.245 - Supplemento Ordinario n.170
- Moore School lectures. (n.d.). Retrieved 01 12, 2016, from https://en.wikipedia.org/wiki/Moore_School_Lectures

Computing Education in a Hybrid World

- Moray, N. (1998). Identifying mental models of complex human-machine systems. *International Journal of Industrial Ergonomics*, 22(4), 293-297.
- Norman, D. (1983). Some observations on mental models. In D. S. Gentner, *Mental Models*. Psychology Press.
- NVAO. (2008). *Nederlands kwalificatieraamwerk Hoger Onderwijs*. NVAO.
- NVAO. (n.d.). *Higher Education Systems in Flanders*. Retrieved 12 28, 2010, from <http://www.nvao.net/higher-education-system-in-flanders>
- NVAO. (n.d.). *Higher Education Systems in the Netherlands*. Retrieved 12 28, 2010, from <http://www.nvao.net/higher-education-system-in-the-netherlands>
- NWO. (2011, 1 5). *Exacte Wetenschappen*. Retrieved from http://www.nwo.nl/nwohome.nsf/pages/NWOP_5S7CLQ
- Object-Relational impedance mismatch, philosophical differences*. (n.d.). Retrieved 11 4, 2013, from http://en.wikipedia.org/wiki/Object-relational_impedance_mismatch#Philosophical_differences
- Open Universiteit. (2012). *Webcultuur*. Heerlen: Open Universiteit.
- Orey, M. (2001). *Emerging perspectives on learning, teaching, and technology*. Retrieved december 13, 2008, from <http://projects.coe.uga.edu/epltt/>
- Pair, C. (1993). Programming, programming languages and programming methods. *Psychology of Programming*, (pp. 9-19).
- Payne, S. (2009). Mental models in human computer interaction. In A. Sears, & J. Jacko, *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications* (pp. 39-52). CRC Press.
- Pennington, N. (1987). Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive psychology*, 19(3), 295-341.
- Plessius, H., & Ravesteyn, P. (2016). Mapping the European e-Competence framework on the domain of Information Technology: a comparative study. *29th Bled eConference - Digital Economy*. Bled.
- Polvinen, E. (2007). Educational Simulations in Second Life for Fashion Technology Students. *Second Life Education Workshop 2007*, (pp. 56-60). Chicago, US.
- Putnam, H. (1975). What is mathematical truth? In H. Putnam, *Mathematics, Matter, and Method* (2nd ed., pp. 60-78). Cambridge: Cambridge University Press.
- QANU. (2007). *Informatica*. Quality Assurance Netherlands Universities (QANU). QANU.
- QANU. (2014). *Informaticaonderwijs aan de Nederlandse Universiteiten in 2013 - state of the art*. Quality Assurance Netherlands Universities (QANU). Utrecht: QANU.
- Ralston, A. (1981). Computer Science, Mathematics and the undergraduate curriculum in both. *Amerian Mathematical Monthly*, 88(7), 472-485.
- Ralston, A., & Shaw, M. (1980). Curriculum '78 is Computer Science really that unmathematical? *Communications of the ACM*, 23(2), pp. 67-70.
- Reilly, E. (2004). *Concise Encyclopedia of Computer Science* (4th ed.). Jphan Wiley & Sons.
- Reisner, P. (1981). Human factors studies of database query languages: A survey and assessment. *ACM computing surveys*, 13(1), 13-31.
- Rijks Universiteit Groningen, Faculty of Arts. (n.d.). Retrieved 3 13, 2011, from Curriculum Informatiekunde: <http://www.rug.nl/let/onderwijs/Bachelor/Informatiekunde>
- Rist, R. (1995). Program Structure and design. *Cognitive science*, 19(4), 507-561.
- Ritzema, T., & Harris, B. (2008). *The use of Second Life for distance education*. Retrieved may 28, 2008, from Nineteenth Annual CCSC South Central Conference: <http://vhil.stanford.edu>
- Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: a review and discussion. *Computer Science Education*, 13, 137-172.

References

- Rovai, A. (2002). Development of an instrument to measure classroom community. *The Internet and Higher Education*, 5(3), 197-211.
- Sahami, M., Danyluk, A., Fincher, S., Fisher, K., Grossman, D., Hawthorne, E., . . . Cuadros-Vargas, E. (2013). *Computer Science Curricula 2013, curriculum guidelines for undergraduate degree programs in Computer Science*. ACM/IEEE.
- Schagen, J., Kwaal, W. v., Leenstra, E., Smit, W., & Vonken, F. (2009). *Bachelor of ICT domeinbeschrijving*. HBO-I stichting.
- Schwank, I. (1993). Cognitive structures and cognitive strategies in algorithmic thinking. In E. Lemut, B. du Boulay, & G. Dettori, *Cognitive models and intelligent environments for learning programming* (Vols. , NATO ASI series F, pp. 249-259). Berlin, Germany: Springer.
- Schwank, I. (1996). Zur Konzeption prädikativer versus funktionaler kognitiver Strukturen und ihrer Anwendung. *ZDM-Analysenheft "Deutsche psychologische Forschung in der Mathematikdidaktik". Zentralblatt für Didaktik der Mathematik*(6), 168-183.
- Schwank, I. (2002). Analysis of eye-movements during functional versus predicative problem solving. *European Research in Mathematics Education II. Selected papers from the 2nd Conference of the European Society for Research in Mathematics Education* (pp. 489-498). Prag: Charles University.
- Sjoer, E., Veeningen, C., Jacobs, F., & de Jong, K. (2008). Lifelong learning in the digital age. *Proceedings of CELDA 2008*. Freiburg, Germany.
- Skilledup. (2016, 5 25). Retrieved from <http://www.skilledup.com/articles/web-developer-job-descriptions-skills-they-require>
- Soloway, E., Jackson, S., Klein, J., Quintana, C., Reed, J., Spitulnik, J., . . . Scala, N. (1997). Learning theory in practice, Case studies of learner-centred Design. *Proceedings of the SIGCHI conference on Human factors in Computing* (pp. 189-196). Vancouver, CN: ACM.
- Soper, D. (n.d.). *Fisher's Exact Test Calculator for a 3X3 Contingency Table*. Retrieved 01 10, 2015, from <http://www.danielsoper.com/statcalc>
- Stanford University. (2001). Retrieved may 28, 2008, from Virtual Human Interaction lab: <http://vhil.stanford.edu>
- Steinkuehler, C. (2004). Learning in Massively Multiplayer Online Games. In I. S. Sciences (Ed.), *6th International Conference on Learning Sciences*, (pp. 521-528). Santa Monica, CA, USA.
- Tartar, J., Arden, B., Booth, T., Denning, P., Miller, R., & van Dam, A. (1984, 5 101-105). 1984 Snowbird Report, Future Issues in Computer Science. *Computer*1985.
- Tedre. (2007). *Lecture Notes in the Philosophy of Computer Science*. Retrieved 9 3, 2015, from <http://cs.joensuu.fi/~mmeri/teaching/2007/philcs/>
- Tedre, M., & Apiola, M. (2013). Three computing traditions in school computing education. In D. M. Kadujevich, C. Angeli, C. Schulte, & Routledge (Ed.), *Improving Computer Science Education* (pp. 100-16).
- UK, e.-S. (n.d.). *The Tech Partnership*. Retrieved february 26, 2018, from <https://www.thetechpartnership.com/about/>
- UNESCO. (2018). *Curricular Aims/Goals*. Retrieved March 15, 2018, from Unesco International Bureau of Education: <http://www.ibe.unesco.org/en/glossary-curriculum-terminology/c/curriculum-aims-goals>
- Valkenburg, M., B. B., Eekhout, M. v., Haperen, M. v., Lousberg-Orbons, A., & Vonken, F. V. (2014). *Domeinbeschrijving Bachelor-ICT*. HBO-I stichting.
- van den Akker, J. (2004). Curriculum perspectives, an introduction. In J. v. Akker, W. Kuiper, & U. Hameyer, *Curriculum landscapes and trends* (pp. 1-10). the Netherlands: Springer.
- van den Akker, J. (2010). Building bridges - how research may improve curriculum policies and classroom practices. In S. M. (ed.), *Beyond Lisbon, 201(0), Perspectives from research*

Computing Education in a Hybrid World

- and development for educational policy in Europe* (pp. 175-195). Sint-Katelijne-Waver, Belgium: CIDREE.
- van der Heide, D. (2002, march 8). Passen en meten. *inaugurele rede Faculteit Geneeskunde*. Universiteit van Maastricht, NL.
- van der Veer, G. (1990). *Learning, individual differences and design recommendations*. Alblasserdam, the Netherlands: Haveka b.v.
- van der Veer, G., & Puerta Melguizo, M. (2002). Mental Models. In *The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications* (pp. 52-80).
- van Roy, P. (2008). *The principal programming paradigms*. Retrieved 11 03, 2013, from <http://www.info.ucl.ac.be/~pvr/paradigmsDIAGRAMeng108.pdf>
- Verifications and Validation of software*. (n.d.). Retrieved from Wikipedia.
- Vivendi. (2004). *World of Warcraft*. Retrieved may 28, 2008, from <http://www.worldofwarcraft.com/index.xml>
- Vlaamse overheid. (n.d.). Retrieved 12 28, 2010, from Het Hoger Onderwijsregister: <http://www.hogeronderwijsregister.be/advanced-search>
- Vyas, D., & van der Veer, G. (2006). Rich evaluations of entertainment experience: bridging the interpretational gap. *13th European Conference on Cognitive Ergonomics*, (pp. 137-144). Zuroch, Switzerland.
- Wages, R., Grünvogel, M., & Grützmacher, B. (2004). How Realistic is Realism? Considerations on the Aesthetics of Computer Games. In M. Rauterberg (Ed.), *Entertainment Computing - Third International Conference*. 3166, pp. 216-225. Heidelberg: Springer.
- Wegner, P. (1970). Three computer traditions: Computer technology, computer mathematics and computer science. *Advances in computers*, 10, 7-78.
- Wegner, P. (1970). Three computer traditions: Computer technology, computer mathematics and computer science. *Advances in computers*, 10, 7-78.
- Wegner, P., & Goldin, D. (2006, July). Principles of problem solving. *Communications of the ACM*, 49(7).
- Wiedenbeck, S., Ramalingam, V., Sarasamma, S., & Corritore, C. (1999). A comparison of the comprehension of object-oriented and procedural programs by novice programmers. *Interacting with computers*, 11(3), 255-282.
- Wing, J. (2006, March). Computational Thinking. *Communications of the ACM*, 49(3), pp. 33-35.
- Yau, S., Ritchie, R., Semon, W., Traub, J., van Dam, A., & Winkler, S. (1983). Meeting the crisis in Computer Science. *Communications of the ACM*, 26(12), 1046-1050.
- Zicari, R. (2012). *Do we still have an impedance mismatch problem?*, interview José A. Blakeley and Rowan Miller. Retrieved 11 17, 2012, from <http://www.odbms.org/blog/2012/05/do-we-still-have-an-impedance-mismatch-problem-interview-with-jose-a-blakeley-and-rowan-miller/>

Summary

Computing is a discipline that is still evolving rapidly. Hybrid professions have emerged, such as medical information systems specialist or specialist in bio informatics; an new curricula have emerged to train these professionals. Should we consider these curricula as computing curricula? Our answer is: partially. We consider some of them as hybrid computing curricula. We consider “hybrid” those curricula that train professionals for a computing-related field, but devote less than 50% of its education to computing. This thesis is a reflection on computing curricula and the computing content of hybrid computing curricula.

In this thesis, we reflect on the education of computing professionals in general and of professionals in hybrid professions in particular. Taking a historical perspective, we describe the origins of the ACM/IEEE Curriculum Recommendation series and comment on it aims. Computing can be approached from different points of view (formal, scientific and design) respecting different epistemological values. The ACM/IEEE Curriculum Recommendation endeavor has a twofold ambition: to train the skilled workforce requested by industry, while supporting the development of one unified discipline of computing. These ambitions are separated in the Dutch system for tertiary education, where responsibility lies with Universities of Applied Sciences and Universities, respectively. To better understand Dutch applied undergraduate computing curricula, we investigate similarities and differences between the Dutch framework for computing education at the Universities of Applied Sciences (HBO-ICT) and international frameworks that were designed to scaffold the design of computing curricula: the ACM/IEEE curriculum recommendation series and the European e-Competence Framework.

We investigate the understanding of abstract concepts by students enrolled in different computing curricula, and find indications for discrepancies. We investigate students’ perception of learning in a hybrid curriculum, and find differences between the results as reported by the lecturers and perceived by the students. Finally, we reflect on the design of a course we have developed for a hybrid audience.

We argue that the curricular ambitions to prepare students for the labor market while supporting the development of one unified discipline of computing, might be incompatible. But separating professional and academic education, as in the Netherlands, also presents unexpected difficulties and pitfalls. We invite national and international organizations to further explore the question of how the three styles of computing relate to

Computing Education in a Hybrid World

the education of computing practitioners. We urge designers of hybrid curricula to take responsibility for the delimitation of new professions, in collaboration with the academic community, the industry, and the public authorities.

We formulate recommendations for undergraduate computing curricula aiming to support conceptualization of knowledge underlying software development skills in a way that (1) prepares graduates to enter the labor market; (2) allows them to keep up with a turbulent profession; and (3) delimits these professions. In doing so, we focus on hybrid curricula in particular.

Samenvatting

De Informatica, die in dit proefschrift wordt aangeduid met de term 'Computing', is een vakgebied dat nog altijd snel in ontwikkeling is. Er ontstaan hybride deelgebieden als medische informatiesystemen of bio-informatica. Nieuwe bachelor-curricula ontstaan om professionals op te leiden in deze deelgebieden. In hoeverre zouden deze moeten worden gerekend tot de Informatica-curricula? Ons antwoord is: ten dele. Sommige daarvan zullen wij 'hybride' noemen. Dat zijn die curricula die bedoeld zijn om professionals op te leiden in Informatica-gerelateerde vakgebieden, terwijl zij minder dan 50% van hun ruimte (studiepunten) wijden aan de Informatica. Dit proefschrift is een beschouwing over Informatica-curricula en het Informaticagedeelte van hybride curricula.

Vanuit een historisch perspectief, beschrijven wij het ontstaan van de ACM/IEEE Curriculum Recommendations Series en bespreken wij hun doelstellingen. De Informatica blijkt te kunnen worden benaderd vanuit drie verschillende invalshoeken (formeel, empirisch-wetenschappelijk en vanuit een ontwerpperspectief), die verschillende epistemologische waarden respecteren. ACM/IEEE heeft zich met de Curriculum Recommendations Series een dubbel doel gesteld: professionals opleiden voor de arbeidsmarkt en de eenheid van het vakgebied bewaken. In Nederland zijn deze ambities respectievelijk ondergebracht bij het HBO en de Universiteiten. Om het Nederlandse systeem beter te begrijpen, brengen wij overeenkomsten en verschillen in kaart tussen de Nederlandse Domeinbeschrijving HBO-Bachelor of ICT en internationale frameworks, die zijn bedoeld om curriculaire keuzes te onderbouwen: de ACM/IEEE curriculum Recommendations series en het Europese e-Competence framework e-CF 3.0.

Wij onderzoeken hoe studenten omgaan met abstracte begrippen. Daarbij vinden wij aanwijzingen voor verschillende conceptualisaties van hetzelfde begrip bij studenten die verschillende programma's hebben gevolgd binnen HBO-ICT. Bij studenten in een hybride curriculum vinden wij een andere perceptie over de leeropbrengsten van een succesvolle cursus, dan wat hun docenten hadden gerapporteerd. Tot slot reflecteren wij over het ontwerp van een cursus over het Web voor een gemengd publiek, bestaande uit eerstejaars studenten Informatica en ouderejaars studenten Cultuurwetenschappen.

Wij concluderen dat de ambities van de ACM/IEEE Curriculum Recommendations series wel eens strijdig zouden kunnen blijken. Maar ook de scheiding van de professionele en de academische variant van het Hoger

Computing Education in a Hybrid World

Onderwijs, zoals in het Nederlandse systeem, kent onverwachte moeilijkheden en valkuilen. Het vraagstuk, wat een solide Informatica-curriculum is in het Hoger Onderwijs, is er een die op dit moment geen pasklare antwoorden kent.

Wij nodigen nationale en internationale organisaties uit om nader te onderzoeken wat de drie invalshoeken betekenen voor de opleiding van Informatica-professionals. Wij dringen er bij ontwerpers van hybride curricula op aan, dat zij verantwoordelijkheid nemen voor de afbakening van de nieuwe deelgebieden van de Informatica waar zij voor opleiden, in samenspraak met de academische gemeenschap, het bedrijfsleven en de overheid.

Tot slot geven wij aanbevelingen voor het ontwerpen van bachelor-curricula in de Informatica, zodanig dat (1) afgestudeerden worden voorbereid op deelname aan de arbeidsmarkt; (2) zij hun turbulente vakgebied kunnen bijhouden; en (3) het deelgebied waarin zij competent zijn is afgebakend. Daarbij richten wij onze aandacht in het bijzonder op hybride curricula.

Curriculum Vitae Laura Benvenuti

2013-now

Lecturer at Hogeschool van Amsterdam (University of Applied Sciences in Amsterdam, NL), Faculty of Digital Media and Creative Industries, bachelor program Communication and Multimedia Design.

2006-2012

Assistant Professor at Open Universiteit Nederland, Faculty of Computer Science.

2003-2006

Lecturer at Hogeschool Utrecht (University of Applied Sciences in Utrecht, NL), bachelor program Digital Communication (successor of the bachelor program Communication Systems - see earlier). The Multimedia studio developed into a specialization program Communication and Multimedia Design. Responsible for the Computing part of this program.

1995-2006

Lecturer at Hogeschool Utrecht, (University of Applied Sciences in Utrecht, NL), Faculty of Communication and Journalism, bachelor in Communication Systems. Responsible for the Computing program until 1999. Later: project manager of a pilot Multimedia studio.

1991 -1993

Trainer/consultant at Remmen & de Brock, Eindhoven (NL), a company specializing in semantic models for relational databases.

1987-1990

Project manager at the Delegation of Italian Central Bank in Brussels (B), pilot office automation.

1986

Master's degree in Mathematics, Mathematical Logic, Dept. of Computer Science, Università degli Studi di Pisa (I).