Self-Beliefs in the Introductory Programming Lab and Game-Based Fantasy Role-Play

Michael James Scott

Department of Computer Science

Brunel University London

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

May 2015

An nescis, mi amicis, quantilla prudentia mundus educatis?

Abstract

It is important for students to engage in adequate *deliberate practice* in order to develop programming expertise. However, students often encounter anxiety when they begin to learn. This can present a challenge to educators because such anxiety can influence practice behaviour. This thesis situates this challenge within the Control-Value Theory of Achievement Emotions, emphasising a need for domain-specific research and presenting new research tools which can be used to investigate the area. Analysis of data collected from three cohorts of introductory programming students on web programming (2011-12) and robot programming (2012-13 and 2013-14) courses show that programming self-concept and programming aptitude mindset can predict programming anxiety and that programming anxiety is negatively correlated with programming practice. However, levels of anxiety remained consistently high across this period. A method to enrich these psychological constructs through a multimedia-rich learning environment is proposed. Drawing upon the interplay between narrative reinforcement and procedural rhetoric that can be achieved in a fantasy role-play, students' self-concept can be enhanced. A double-blind randomised controlled trial demonstrates promising results, however small effect sizes suggest further research is needed.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my supervisor, Gheorghita Ghinea, for his patience and support throughout the doctoral programme.

Declaration

I declare that this thesis is an account of an original research. The following articles have been produced as a direct or indirect result of the research discussed in this thesis:

Journal Articles:

1. Scott, M. J. and Ghinea, G., Enriching the Self-Concept and Mindset of Novice Programmers using Game-based Fantasy Role-Play: A Review of Existing Games, *British Journal of Educational Technology* (Under Review), pp. 1–6, DOI: 10.1109/TE.2013.2288700.
2. Scott, M. J., Counsell, S., Lauria, S., Swift, S., Tucker, A., Shepperd, M., and Ghinea, G., Enhancing Practice and Achievement in Introductory Programming with a Robot Olympics, *IEEE Transactions on Education* (Accepted, In Press), pp. 1–6, DOI: 10.1109/TE.2013.2288700.
3. Scott, M. J. and Ghinea, G., On the Domain-Specificity of Mindsets: The Relationship Between Aptitude Beliefs and Programming Practice, *IEEE Transactions on Education*, 57 (2014), pp. 169–174, DOI: 10.1109/TE.2013.2288700.

Peer-Reviewed Papers:

4. Scott, M. J. and Ghinea, G., Measuring Enrichment: The Assembly and Validation of an Instrument to Assess Student Self-beliefs in CS1, *Proceedings of the 10th Annual ACM Conference on International Computing Education Research (Glasgow, Scotland)*, August 2014, pp. 123–130.
5. Scott, M. J. and Ghinea, G., Implicit Theories of Programming Aptitude as a Barrier to Learning to Code: Are They Distinct from Intelligence?, *Proceedings of the 18th ACM Annual Conference on Innovation and Technology in Computer Science Education (Kent, UK)*, July 2013, p. 347.
6. Scott, M. J., Projective Identity and Procedural Rhetoric in Educational Multimedia: Towards the Enrichment of Programming Self-concept and Growth Mindset with narrative reinforcement, *Proceedings of the 21st ACM International Conference on Multimedia (Barcelona, Spain)*, October 2013, pp. 1031–1034.
7. Scott, M. J. and Ghinea, G., Educating Programmers: A Reflection on Barriers to Deliberate Practice, *Proceedings of the 2nd HEA STEM Conference (Birmingham, UK)*, April 2013, pp. 28–33.
8. Scott, M. J. and Ghinea, G., Integrating narrative reinforcement into the Programming Lab: Exploring the 'Projective Identity'

Hypothesis, *Proceedings of 43rd ACM Technical Symposium on Computer Science Education (Denver, CO, USA)*, March 2013, pp. 119–122.

1

Introduction

Programming is an area of challenge within computing education research. This may be because the progress of some students may be impeded by their emotions or by their self-beliefs. However, little research shows how such phenomena arise and can be overcome. Therefore, it is important to explore how self-beliefs and emotions interact with the way students learn programming. This chapter forms the foundation for this exploration, outlining the research in this thesis.

1.1 Overview

Educational multimedia is widely used to support learning. Applications include: instructional videos; visualizations; training simulations; and serious games. Part of its popularity can be attributed to its availability, as learners in many parts of the world are able to access multimedia-enabled instruction due to the growing ubiquity of computer-based tools [MR01]. Furthermore, multimedia communication is often informed by the principles of cognitive science, so the delivery of learning material in such cases is optimized for human information processing [May08, MCGC11].

However, while retention and transfer of knowledge are highly desirable, direct instruction is not the only goal of education. At the tertiary level, in particular, there is a movement towards learner-centred paradigms which champion autonomy [BT95] and encourage self-regulation [Zim02]. This has implications for practical disciplines, such as computer programming, where it is desirable for learners to immerse themselves in a regime of ongoing reflective practice. This is because at least ten years of such *deliberate practice* is often needed to obtain substantial expertise [EKTR93**?**].

Yet, despite this renewed emphasis, educators do not appear to have overcome some of the most pervasive challenges in introductory programming [Guz11, TG11, SBE83, MAD$^+$01]. Many novices do not appear to practice programming regularly, claiming they experience apprehension and discomfort when they attempt to do so [KS10b, RS10b]. Some authors even go so far as to describe this as "programming trauma" [Hug04], and there is some neurological evidence in the area of mathematics education which suggests such anxiety is linked with regions of the brain responsible for visceral threat detection and pain [LB12]. Thus, negative experiences and their impact on avoidance behaviour is an area of concern for computing educators.

The potential barriers that influence programming avoidance are numerous and multifaceted (see [BM05, Bor11]). As such, it is important to incorporate analytical and adaptable approaches into the design of learning environments and, in particular, in the design of tools that educators use to support these learning environments. These then permit educators to diagose and solve challenges associated with these barriers.

Unfortunately, many diagnostic processes and solutions do not trivially scale to large cohorts of students that many educators are responsible for. Moreover, the computing educational research literature does not clearly articulate the pedagogic-content knowledge that educators could use to support individual programming students more effectively (not to be confused with best practices in instructional design for which much research exists, e.g., [PS13]).

The means through which these diagnostic techniques and educational interventions are applied in practice, nevertheless, does not rely solely on teaching assistants and faculty. Educational multimedia and adaptive hypermedia systems can be used to support scalability. Therefore, suitable measurement, scaffolding and feedback for individual learners can be managed by faculty with the support of virtual learning environments. This enables pedagogies that account for barriers to deliberate practice.

1.2 Aims and Objectives

The aims of this thesis is twofold: firstly, to understand the relationship between self-beliefs, emotions, and practice behaviour in the introductory programming context; and secondly, to understand how to support students' ability to overcome non-constructive self-beliefs, to incorporate positive emotions into learning experiences, and thereby how to improve student practice. As such, the thesis is broadly separated into four key areas:

- Constructing a conceptual framework which can be used to understand the relationship between self-beliefs, emotions, and practice
- Developing valid measurement instruments which can be used to investigate these relationships
- Assessing the impact of existing teaching practices in the introductory programming lab on self-beliefs and emotions
- Performing experimental research which investigates how to overcome non-constructive self-beliefs using new teaching tools

Based on these four key activities, a range of contributions is expected. This includes the development of research tools (also referred to as "instantiations") which can be used as a lens for further research into self-beliefs and emotions in the introductory programming context. This leads into using these new research tools to support the development and assessment of new pedagogical tools with self-belief and emotion requirements, resulting in recommendations which could be incorporated into the development of future pedagogical tools.

1.3 Philosophy and Approach

This thesis follows the pragmatic philosophy for conducting research in education. This is centred on a perspective of research which privileges the exploration of applied contexts and practical utility over theoretical impact. It also emphasises the ontological position of transactional realism; that is, the notion that reality exists within ephemeral interaction between organisms and their environment.

These are, as argued in the methodology chapter, systematic. However, the evolutionary nature of both organisms and environments can present challenges to external validity. As such, the research discussed in this thesis focuses on the development of conceptual models which educators can use to diagnose and understand problems in their learning environment and then proposes some potential solutions which educators can use to overcome the challenges they face.

As much valuable qualitative work has already been done in the area of self-beliefs and emotion, the research contained in this thesis focuses on quantitative approaches to measuring self-beliefs and emotion. This offers validity and verification to work already conducted in this area and reveals the significance of the problem in a way that non-representative samples favoured in qualitative research cannot.

1.4 Structure of the Thesis

This chapter introduces educational multimedia as a key driver in the success of learning environments. While the motivational characteristics of such multimedia is widely studied, both in terms of cognition and affect, opportunities for further research on its role in preparing students for self-regulated learning are highlighted. It then illustrates introductory computer programming as an area of challenge in this respect. The reasons why some students seem to develop negative emotions, such as anxiety, during their first programming course was questioned. Several research questions associated with the way educators conceive emotion, practice, and achievement were raised, leading to the hypotheses associated with the role of educational multimedia in enriching these self-beliefs.

The literature review presented in *Chapter Two* discusses the challenges associated with the introductory programming lab in further detail, focusing on complex relationship between belief, emotion, and behaviour. The Control-Value Theory of Achievement Emotions is introduced as a means to understand these challenges and how to help individual students overcome barriers to deliberate practice. It is posited, however, that the theoretic framework may not be transferable (at least, in a trivial manner) to the programming context and may require adaptation through the introduction of domain-specific constructs, with the findings of the review demonstrating that few such explorations have been conducted.

Chapter Three discusses the pragmatic philosophy which underpins the research described in this thesis. The process of *Pragmatic Design Research*, as a process for computing education research, is then described alongside further detail on its ontological, epistemological, and axiological underpinnings. Then, after emphasising the value of the applied contributions this approach makes to the displicine, the chapter proposes a strategy based on: model development; design; and experimental evaluation.

The study presented in *Chapter Four* reinforces the rationale for the research raised in Chapter Two by demonstrating that a domain-specific approach to self-belief research can be justified within the computer programming context. The results of a longitudinal survey supports the hypothesis that a domain-specific mindset construct (mindset towards programming aptitude) can, when compared to a more general mindset construct (mindset towards intelligence), have greater utility for predicting programming behaviours.

Chapter Five builds upon the work in the previous chapter, introducing a measurement instrument which has been adapted for use within the domain of introductory programming in the higher education context. In doing so, the chapter also presents a number of challenges associated with valid and reliable measurement. In particular, the importance of validating how psychological constructs are measured. Thereby building upon the lack of validation research within the computing education research field through demonstrating one approach to demonstrating adequate validity.

The analysis presented in *Chapter Six* evaluates the self-beliefs, emotions, practice, and performance of two cohorts of students. Two course designs are compared: a robot-centred course design used in 2012-13 cohort and the web programming course design used in 2011-12. Despite some improvements in student outcomes, there were no significant differences between the two cohorts in terms of self-beliefs or emotion. Nevertheless, relationships between programming self-concept, programming aptitude mindset, programming anxiety, programming practice, and three aspects of program quality (i.e., functional coherence, readability, and sophistication) are observed, suggesting that the enrichment of student self-beliefs could be a beneficial pedagogic strategy.

As the robots did not have a significant impact on students' beliefs or emotions, alternative strategies need to be considered. As such *Chapter Seven* presents a literature review on the use of fantasy role-play to support the introductory programming lab. Namely, narrative reinforcement and procedural rhetoric. The potential benefits of game-based approaches using fantasy role-play are discussed in relation to pedagogical theory and then a review of over 50 programming games is presented. Despite the relatively large number of games and their diversity in design, few seemed to take advantage of the proposed techniques and none had been empirically evaluated in terms of student self-beliefs.

This review revealed little empirical support for the transformation effects of games claimed by serious game evangelists in the computing education field. Hence, the study presented in *Chapter Eight* empirically evaluates the hypothesis that a projective identity can be used to enhance students' self-beliefs. Specifically, programming self-concept. The chapter discusses the theories associated with the notion of projective identities, focusing on the Proteus Effect, and describes the development of two experimental e-learning activities: one integrating fantasy role-play and the other without. The results of an experimental trial are promising but did not suggest a practically meaningful effect.

Chapter Nine discusses the overall findings of the research and how they relate to the aims and objectives presented in Chapter One while emphasising the contributions made to the fields of computing education research and educational technology. As with all research, this work has limitations which the chapter acknowledges, suggesting future work which could be conducted to overcome these limitations or otherwise improve the validity of the findings. In particular, opportunities for further development and longitudinal evaluation are considered.

Declaration

Some of the work presented in this chapter can also be found in the following publication:

Scott, M. J., Projective Identity and Procedural Rhetoric in Educational Multimedia: Towards the Enrichment of Programming Self-concept and Growth Mindset with narrative reinforcement, *Proceedings of the 21st ACM International Conference on Multimedia (Barcelona, Spain)*, October 2013, pp. 1031–1034

2

Challenges in the Introductory Programming Lab

This chapter reviews a number of the challenges that educators often encounter in the introductory programming lab. It examines the importance of deliberate practice and the role that emotions play, anxiety in particular, in inhibiting this form of practice. A conceptual framework based on the Control-Value Theory of Achievement Emotions is presented and the potential use of the framework to develop new pedagogical approaches is discussed.

2.1 Introduction

In recent years, there has been a drive to revitalise computing education in the United Kingdom [BSCH14], in part due to criticisms by the Royal Society [Fur12], the Nesta Trust [LH11] as well as various other professional and government bodies [PJM10, Gov12]. The new curriculum introduced in 2014 includes the fundamentals of computer programming [fE13]. However, programming can be very challenging to teach [Jen02]. Subsequently, there are concerns that inexperienced educators will require support in this endeavour [LH11].

While such concerns are not unfounded, it is important to note that even students taught by those with substantial expertise often fail to progress to more advanced courses. Notwithstanding those which have shown improvements in retention and success [PS13], many studies conducted across the last 30 years show poor outcomes at the introductory level [Guz11, TG11, SBE83, MAD+01]. In addition, while failure is high [BC07], many successful students *choose* not to progress [BM05, Car06b]. The reasons for these outcomes are complex and multifaceted (see [BM05]). Hence, introductory programming is considered a grand challenge in computing education [MBI+05].

In line with the theory that it can take approximately ten years to become an expert in software development [**?**], the discipline demands a substantial level of *deliberate practice* in order to achieve mastery [EKTR93**?**]. As such, a key issue can sometimes be both the amount as well as the *quality* of practice that novice programmers engage in [**?**]. This means that practice needs to be ongoing, focused, reflective, and situated at the right level of challenge for any individual student [EKTR93]. This is supported by evidence within the field of computing education research which suggests that levels of effort [VJ05], comfort [WS01, VJ05] and depth [SFS+06] can predict success in the first programming course. This type of practice, however, is inherently uncomfortable and demands that learners remain focused and motivated.

Unfortunately, few beginners appear to find writing code easy, with even fewer being able to maintain their focus and motivation [Jen01, Jen02]. As such, crafting a learning environment which promotes deliberate practice is not a trivial task. One of the challenges associated with these problems is the level of doubt, fear and anxiety that students experience as they engage with programming tasks [KS10b, RS10b]. Despite the best efforts of instructors, learners still report negative experiences when they engage with programming tasks [KB12, RS10a]. Some authors describe this phenomenon as *programming trauma* [Hug04] and, to reinforce such striking language, there is some evidence which indicates that the type of task anxiety these experiences invoke are related to the activation of brain regions associated with visceral threat detection and pain [LB12]. Of further concern, empirical work has shown that such affective states tend to worsen across the duration of introductory courses [MD04].

It is generally accepted that reducing anxiety can enhance academic performance [Hat09]. So, to alleviate anxiety, educators often work closely with students to provide support [Jen01]). However, it is important that educators intervene to address causes rather than symptoms. It is possible that some of the anxieties that students experience could be reflective of underlying self-beliefs which may be non-constructive. As Pajares points out in a review on beliefs in education: "beliefs are the best indicators of the decisions individuals make throughout their lives " [Paj92, p. 307]. Naturally, it follows that beliefs can inform decisions about instructional and learning strategies [SS80].

The self-beliefs which students develop appear to manifest as a result of the experiences students have while they engage in programming activities rather than the resulting quality of the programs they write [KS12]. With this being the case, the emotions that learners feel may prompt them to reflect on themselves and their ability in several different ways [KB12]. Potentially, learners may start to believe that they no longer have the time or the motivation to overcome these challenges as they cannot envision success in the future [KM06]. In other words, learners may change their self-beliefs based on their experiences, through a process of self-appraisal, potentially diminishing the way that they identify with programming as a discipline and disengaging with deliberate forms of practice [PP13**?**]. A consequence of this is that learners claim that they lack time or have no motivation (as in [KM06]). So, if deliberate practice is a key element in the acquisition of programming competencies, how do educators create learning environments that successfully encourage practice?

2.2 Barriers to Programming Practice

In order to appreciate how to facilitate frequent practice, the barriers that prevent it should be explored. Programming is markedly distinct of other disciplines because proficiency in other areas does not predict success [BL01, EAK08] and some believe that there are no effective aptitude tests [MBI$^+$05, CLB07], assuming that aptitudes for programming even exist [EKTR93, Jen02, AL13]. This is because the learning material sometimes demands something very novel to new learners, drawing on skills that, at present, are seldom developed prior to programming instruction:

> By means of metaphors and analogies we try to link the new to the old, the novel to the familiar. Under sufficiently slow and gradual

> change, it works reasonably well; in the case of a sharp discontinuity, however, the method breaks down. [Dij89, p. 1398]

The sudden sense of “radical novelty” [Dij89] creates an unexpected challenge for many learners, presenting a barrier to learning. This is because those without prior experience need to adapt to thinking about the intangible and abstract concepts which are needed to describe the mechanics behind the code they are writing [Bou86]. Barriers can even arise as early as the first stage of instruction. Consider how someone new to reading program code might conceive the mechanics behind an assignment operation, such as:

```
a = 1;
b = 2;
a = b;
```

What is the value of a?

Bornat, Dehnadi and Simon found that for “simple” assignment operations that “hardly look as if they should be hurdles at all” [BDS08, p. 54], students held many different mental models for how the program may execute. Even after a few weeks of instruction, some participants failed to apply the correct model consistently in a diagnostic test. This illustrates that the ways in which learners conceptualise computer programs can be diverse and incorrect models may persist without some intervention. Consequently, it is important not to dismiss the early challenges experienced by individuals as: trivial; a lack of effort; or a lack of talent. Put elegantly, “if students struggle to learn something, it follows that this is for some reason difficult to learn" [Jen01, p. 53]. These issues can be addressed through soft scaffolding, such that individual understandings are continuously probed to enable the timely delivery of tailored support [SK07a]. Through this, misunderstandings are traced and corrected through the provision of intermediate learning objectives. When not promptly addressed, such issues can impede progress as learners are forced to the edge of, or perhaps beyond, their "zone of proximal development" [Vyg80, p. 86]. Because building mental models is easier when extending from a firm foundation this can, of course, lead to significant variation in the types of situation that students encounter and the speed at which they master the material; a concept known as "learning edge momentum" [Rob12, Rob10]. Unfortunately, this could be one (of many) drivers behind the apparent bi-modal distributions of ability (see [DB06] and [Bor14] for an addendum) which reinforce non-constructive beliefs that teachers hold about their students and that students may subsequently develop:

> If teachers wish to create lasting improvements in the performances of low ability students, it may not be enough for them to simply adopt teaching strategies that favour such students. If teachers fail to revise their evaluations of low ability students or the students themselves fail to modify any negative self-evaluations they may have, such students may later revert to their former performance levels. From this perspective, it becomes easier to understand why so many (often erroneous) social stereotypes and idiosyncratic social perceptions are so resistant to change [..] For even if individuals adopt

> interaction strategies that produce behavioural dis-confirmation, their insensitivity to dis-confirmatory information and their tendency to communicate their expectancies to the targets of their beliefs may insure that their beliefs ultimately will receive behavioural confirmation. [SS80, p. 887]

Nevertheless, Kinnunen and Malmi note there can be "individual variety in how students respond to the same situation" [KM06, p. 107]. Many learners who encounter such challenges are able to overcome them without assistance, albeit perhaps after some frustration. So why are some people tenacious while others seem helpless? A potential candidate for mediating this response is an individual's self-beliefs. Notably, implicit beliefs surrounding programming aptitude. Dweck divides learners into entity-theorists, whom believe their aptitude is a natural fixed trait, and incremental-theorists, whom believe their aptitude is a malleable quality which is increased through effort [Dwe99]. These two groups demonstrate different behaviours when they encounter difficulty as summarised in Table 2.1.

Too often, it is the case that learners start to believe an inherent aptitude is required to become a programmer. Such beliefs inhibit practice. Thus, it is important that programming pedagogies reinforce the incremental theory. An example might include the liberal use of detailed informative feedback. This approach focuses on improvement through illustrating weaknesses to overcome, rather than merely labelling learners with summative grades. The latter might be interpreted as a judgement of aptitude. However, many learners "often focus on topics associated with assessment and nothing else" [GS04, p. 14] so some form of formality is often necessary as an extrinsic motivator.

	Entity-Theorists	**Incremental-Theorists**
Goal of the student?	To demonstrate a high coding ability.	To improve coding ability, even if reveals poor progress.
Meaning of failure?	Indicator of low programming aptitude.	Indicative of lack of effort, strategy, or pre-requisites.
Meaning of effort?	Demonstrates low programming aptitude.	Method of enhancing programming aptitude.
Challenge strategy?	Less time practising.	More time practising.
Performance after failure?	Impaired.	Equal or improved.

Table 2.1: The potential influences of those learners with different implicit conceptions of programming aptitude (Adapted from [Dwe99]).

While Dweck's dichotomy is useful in illustrating some differences [Dwe99], it does not explain why some learners seem far more determined than others. Potential factors, as Rogerson and Scott affirm [RS10a], are the negative affective states that learners can experience as they write code. These "states[,] such as frustration and anxiety[, can] impede progress toward learning goals" [MLL07, p. 698]. However, while some learners become overtly frustrated with the all or nothing nature of preparing a computer program for compilation, others press on without complaint, demonstrating an admirable level of experimentation and debugging proficiency. This can be somewhat surprising given that anything short of a completely syntactically correct set of coded instructions will result in failure and it is unusual for those at an introductory level to write robust code on their first attempt.

A potential candidate for mediating how learners are able to overcome negative affect is academic self-concept. That is, “self-perceptions formed through experience with and interpretations of one’s environment” [MM11, p. 60]. It is important to note that its effects are domain-specific [Hua11, RAB12] and so therefore further work in the the computing domain is needed, but it has demonstrated a reciprocal relationship with general academic achievement [MM11] as well as interactions with a range of study-related emotions [GCF+10]. This suggests that learners who believe that they will definitely be able to program, those with a high academic self-concept in programming, are better able to overcome frustrations and anxiety. Thus, they are able to maintain high levels of motivation. However, how can self-concept be enhanced? A meta-analysis of 200 interventions shows that practices which target a domain-specific facet of self-concept, with an emphasis on motivational praise and feedback alongside skill development, yield the largest effects [OMCD06]. Other aspects of effective practice might also place emphasis on learning activities that attempt to nurture senses of pride, enabling high levels of enjoyment [GCF+10].

2.3 The Control-Value Theory of Achievement Emotions

A framework that considers the role of self-beliefs and emotions in learning is the Control-Value Theory of Achievement Emotion [Pek06, PS10]. In this framework, students’ self-appraisal of ongoing achievement activities, and of their past and future outcomes, are of key importance. This is because the emotions that they experience during a particular task will depend upon whether they feel in control of the outcome and that the outcome is subjectively important to them.

These emotions then influence academic engagement and performance through the model shown in Figure 2.1. The model proposes that instruction and support have an influence on the way in which individuals form the control and value appraisals. These appraisals then shape the specific achievement emotions that students may experience based on whether they feel they can control activities and outcomes that are subjectively important to them. These emotions then have a direct impact on self-regulated learning and performance. Specifically, emotions seem to influence cognitive resources, use of strategies, and dependence on external regulation of learning [Pek06]. The overall model is also reciprocal in nature, such that outcomes can shape emotions while both emotion and performance shape the way students form their self-appraisals. In some cases, instruction and support may also respond to student needs. In particular, offering a range of interventions which could influence any part of the model. As this process continues over time, it could have substantial impact on learning behaviour and subsequently performance; as evidenced through the known co-variance between self-efficacy beliefs and success [Wie05, WS01].

As each component of the framework represent a broad range of different constructs, a parsimonious conceptual framework has been embedded within the model. This example is derived from the challenges hypothesised to influence programming practice in the previous section and illustrates how learning activities and feedback influence students' self-beliefs. Namely: self-concept, which is is understood to be a composite of "self-perceptions that are formed

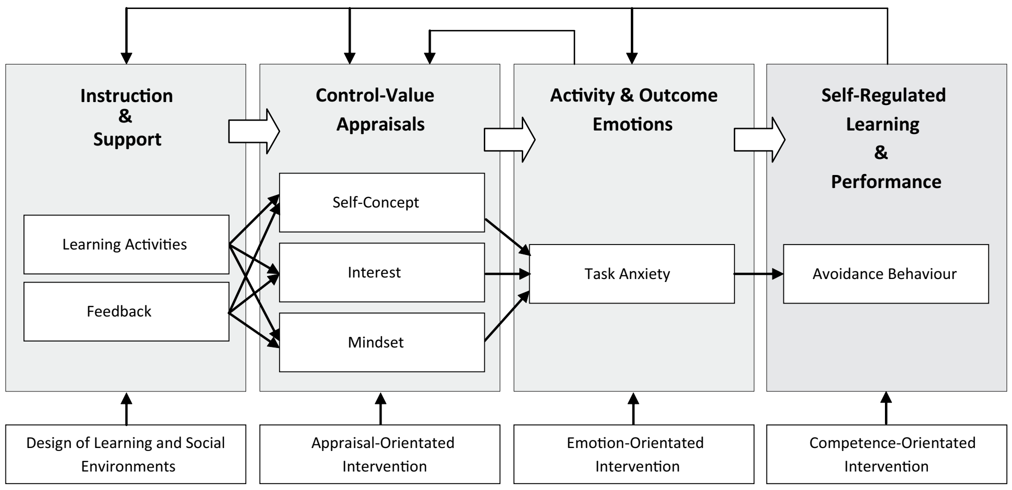

Instruction & Support
Control-Value Appraisals
Activity & Outcome Emotions
Self-Regulated Learning & Performance
Learning Activities
Feedback
Self-Concept
Interest
Mindset
Task Anxiety
Avoidance Behaviour
Design of Learning and Social Environments
Appraisal-Orientated Intervention
Emotion-Orientated Intervention
Competence-Orientated Intervention

Figure 2.1: Overview of the Control-Value Theory of Achievement Emotion with an Embedded Conceptual Framework (Adapted From [Pek06, PS10])

through experience with and interpretations of one's environment" [MM11]; interest, which is the extent to which an individual enjoys engaging with a set of tasks; and mindset, using Dweck's [Dwe99] notion of mindsets which claims that students either have a growth mindset (i.e., they believe their capacities can be developed through practice) or students have a fixed mindset (i.e., they believe their capacities are natural, inherent qualities). These, in turn, influence task anxiety which, consequently, may encourage avoidance behaviour.

2.4 Need for Further Research

Previous research reviews do not discriminate between cognitive and affective goals in education research (see, .e.g., [Val04, MSS$^+$10]). Indeed, some authors conclude that the literature reflects "higher interest in determining what the students can do rather than their behavioural or affective responses to their learning or teaching experience" [SSHL09, p. 101]. This has been reinforced in a recent review of theoretical frameworks which revealed few psychological frameworks beyond Bandura's self-efficacy have been used by researchers [MSS+14].

Historically, the field has a rich history of developing tools to support programming education. This is particularly the case with the ACM's ITiCSE and SIGCSE conferences. Many papers focus on the dissemination of new tools to: automate assessment; automate feedback; reduce cognitive load; improve the usability of introductory programming environments; and many more. However, the evaluation of these tools is typically limited to the particular problem they are designed to solve without addressing their impact more holistically (e.g. how the change in learning environment influences students' self-beliefs and emotions).

The Control-Value Theory of Achievement Emotions [Pek06, PS10] poses causal links between self-beliefs, emotion, and action. Therefore, it would seems pertinent to enrich these self-beliefs in order to help students manage their anxiety and subsequently help them to engage in deliberate practice regularly. Yet, with one noted exception of the Cutts experiment ([CCD+10], there has been very little research on interventions in this area. Thus, this affective dimension of learning computer programming requires attention.

There are a number of practices which could be applied to address the affective domain. For example, growth messages can be embedded into assessment rubrics and feedback [CCD+10], soft-scaffolding can help students to develop a stronger self-concept [OMCD06], and students can be encouraged to engage in work that they can take pride in [GCF+10]. There are, however, some limitations associated with these approaches. In particular, educators in the UK often now need to manage the scalability of their teaching responsibilities due to increasing administrative demands [Tig10] and increasing staff-student ratios [Cou12].

As such, contact-time with students is often constrained [Cou12] *in de-facto lieu* of increased workloads and stress [Kin01]. Hence, scalability is addressed with the use of educational tools. However, these may not be suited to providing the necessary types of tasks and feedback that will enhance students' self-beliefs. In some institutions, this may be supplemented by the use of teaching assistants who conduct reviews of student work. However, it is not clear whether they receive the necessary preparation needed to implement the types of interventions being proposed. An alternative approach would, therefore, be welcome in large classes. A means to achieve this would be, following tradition in the field, to embed interventions directly into educational tools and one type of tool that shows promise, in this respect, is digital games.

2.5 Summary

Learners often need to practice writing code frequently in order to acquire basic programming competencies. This paper questions how learning environments can be better designed in order to facilitate deliberate practice, describing three potential barriers to deliberate practice: the radical novelty of the learning material; the belief that some inherent aptitude is required; and the emergence of unfavourable affective states. Overcoming such barriers will facilitate educators in aiming for excellence, but often require strategies that are analytical and adaptable. It is proposed that examples of good practice include: soft scaffolding; ongoing detailed informative feedback; and an emphasis on self-enhancement, through motivational feedback and pride-worthy activities, in addition to skill-development.

Declaration

Some of the work presented in this chapter can also be found in the following publication:

Scott, M. J. and Ghinea, G., Educating Programmers: A Reflection on Barriers to Deliberate Practice, *Proceedings of the 2nd HEA STEM Conference (Birmingham, UK)*, April 2013, pp. 28–33

3

Computing Education Research: A Pragmatic Design Approach

Computing education research lacks a dominant paradigm that defines its approach to research. This chapter, therefore, discusses the plurality of approaches and traditions adopted within the field, proposing a 'twin-spiral' model of pragmatic design research. It is argued that the model is appropriate for the exploration of the problems and the conceptual framework outlined in Chapter 2. In doing so, it describes the philosophical underpinnings, methodological choices, and research strategy for the work discussed in this thesis.

3.1 Introduction

The previous chapter highlights the role of self-beliefs when learning computer programming and presents a conceptual model centred upon two specific aspects of self-belief, posing a relationship with anxiety and practice. The question raised, however, is whether the proposed model holds and, if so, how it can be used to develop new educational interventions, with Chapter 2 demonstrating opportunities for further research in this area. The work discussed in this thesis, therefore, firstly aims to alleviate this gap by developing and verifying the utility of the proposed conceptual model. Secondly, it aims to explore how new practices and tools could be used to overcome non-constructive self-beliefs. However, the most appropriate approach for this undertaking is unclear.

This chapter addresses the question: what approach to research is appropriate for addressing the challenges presented by the emergence of non-constructive self-beliefs in introductory programming? The following sections will discuss the role of scholarship in computing education research, leading to a discussion on its aspirations and the challenges associated with producing high-quality research under the constraints that novice computing education researchers experience. The next section proposes a research approach which aims to address these aspirations. This includes a review of its philosophical underpinnings, its methodological choice, its research strategy, and the author's preference for data collection and analysis techniques. The chapter then closes with a brief summary of how the approach has been adopted throughout the thesis, its success, and its practical limitations.

3.2 Scholarship in Computing Education

It is important to discuss approaches to scholarship. Its exploration reveals the relationship between an author's work with that of peers within the same field of interest, exposing a variety of different—sometimes, conflicting—world-views and biases associated with their origin. Computing education research, in particular, is characterised as both an interdisciplinary field, owing to its broad and diverse base from which it synthesises knowledge [TBC12, MSS+10], and an emerging discipline, owing to its comparatively short existence as a self-contained area and its lack of widely-adopted theories and research paradigms [MSS+14]. As such, there is considerable plurality that researchers can draw from when developing their approach to research.

3.2.1 The Science of the Artificial

Education research deals with the development of learning and the learning experience. Its goal is the enhancement and enrichment of educational practice. Subsequently, it is not just natural and behavioural science, which focuses on explaining what natural and social phenomena are and how they work [Sim96]. There are, indeed, elements of natural science which are useful as the basis for achieving this aim. However, education is an social institution with many artificial elements which are manipulated by its many stakeholders. As such, the research is quite different. Doctoral studies in this area tend to be concerned with instructional design as well as the design of learning environments and tools [Mal13, Mal14]. In other words, computing education research is concerned with the architecture of education — which could, broadly speaking, be considered the *science of the artificial*, a form of research that focuses on how to design and implement artificial systems and artefacts [Car06a, MS95].

This implies two key activities that form research activity: (i) construction of artificial systems and artefacts; and (ii) the evaluation of artificial systems and artefacts in order to determine whether or not progress has been made

[MS95]. A review of the literature has claimed, guardedly, that the approach demonstrates utility [AS12]. However, the goals of the research and the means to achieve it can be synthesised using a wide variety of other research activities. As such, this approach to research complements traditional research activities and discourse (e.g. from psychology, neuroscience, engineering, etc.). The underlying philosophical perspective would also seem to permit a range of methods, while being suitable for this area of research and the specific challenge the author aims to address. Before describing the approach at this deeper level, however, it is first necessary to discuss the nature of research activities and their respective contributions.

3.2.2 The Boyer Model of Scholarship

At a basic level, adapting the work of Ernest Boyer [Boy90], there are three key forms of scholarship in computing education: scholarship of *discovery*; scholarship of *application*; and scholarship of *integration*. Discovery is what many would consider original research activity, that which aims to contribute new knowledge to the field. Such activities often form a new understanding of a problem, a new pedagogical approach, or a new way of thinking about theory. Application is the analysis of engagement in practice. For example, a promising pedagogical approach is trialled and evaluated in a practical setting. Integration, of course, is the act of drawing together research findings and giving them new perspectives. Typically, this takes the form of a literature review or a meta-analysis.

It is important to recognise that knowledge is not first discovered and then applied. Both forms of scholarship have a dynamic relationship. For example, the findings from an application can lead to new conjectures and hypotheses which inform future discovery. Overall, then, scholarship within computing education research could be conceived as a duality between discovery and application such that scholars develop and apply new practice so that "theory and practice vitally interact, and one renews the other" [Boy90, p. 23].

It should also be noted that scholarship of integration is a constant process where: (i) existing knowledge is synthesised from the literature and applied in research activity; (ii) subsequently, the new knowledge derived from those activities are disseminated to and integrated into the literature; and (iii) literature is drawn together, synthesised, and presented again with new insights.

3.2.3 The Empirical Approach

While the scholarship of integration is, by its nature, theoretic in nature, scholarship of application and discovery tend to privilege empirical evidence in their presentation and defence. This is because our access to reality is through our senses and so sensory data which can be independently verified and replicated is needed to support claims made by researchers in the field [Cha13]. As such, many contributions to the computing education research discipline use an approach to research which relies on systematic observation [Mal13, Mal14]. Such systematic collections of observations are typically analysed and interpreted to determine the conclusion. The process is shown in Figure 3.1.

While the diagram seems to show a closed circle, starting with a problem and ends with a problem, it does not. Rather, it begins a series of spirals. This is because new problems often arise out of previous research. However, these

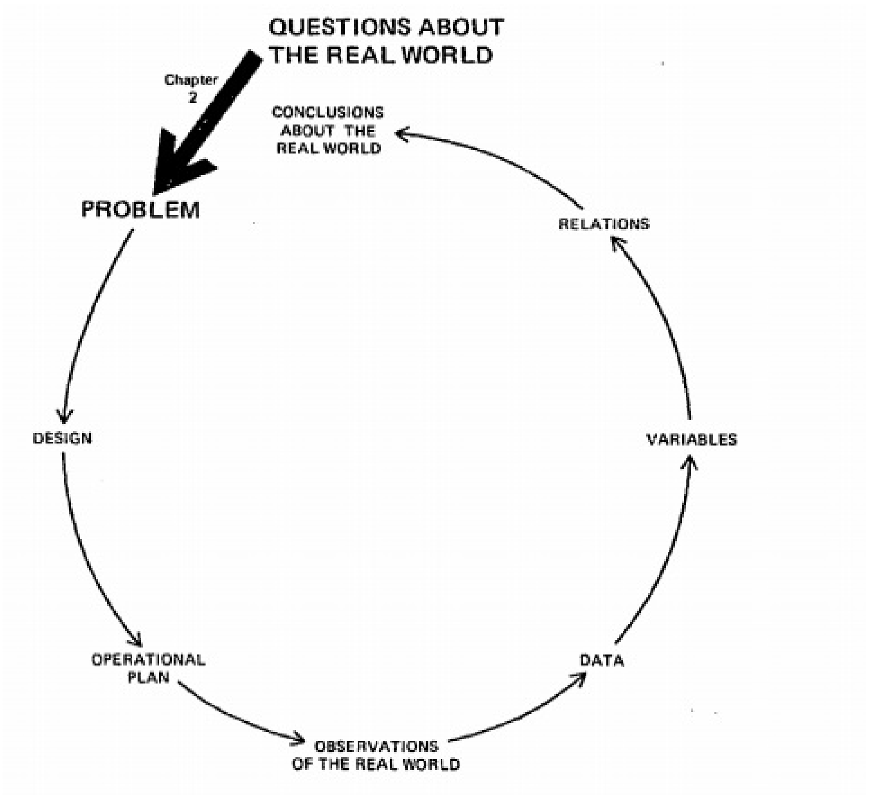

QUESTIONS ABOUT THE REAL WORLD
Chapter 2
PROBLEM
DESIGN
OPERATIONAL PLAN
OBSERVATIONS OF THE REAL WORLD
DATA
VARIABLES
RELATIONS
CONCLUSIONS ABOUT THE REAL WORLD

Figure 3.1: The Cycle of Empirical Research from [McG81, p. 181]

problems may not necessitate the same approach and some conclusions may lead to further work with broader claims.

3.2.4 Aspirations in Computing Education Research

Given that educational research is a design science that aims to improve an artificial architecture and consists of multiple activities—mostly, following the empirical approach—what are the products of the research activity and how can they be judged? According to March & Smith [MS95], there are four types of research output: constructs; models; methods; and instantiations. Constructs form the vocabulary for a domain, constituting the conceptions used to design problems and specify their solutions. Models are sets of propositions expressing relationships between constructs, often representing situations or laws from which problems are articulated and solutions are derived. Methods are synonymous with algorithms, such that they are guidelines for achieving a particular outcome. An instantiation is the realisation of an artefact. In the area of computing education research, this includes both research tools, such as data collection instruments, and educational tools, often some form of software.

While the products of this approach to research are varied, they share a set of common characteristics from which quality can be judged and aspired to. While quality is difficult to define and quantify, as the outputs are artificial in nature, quality should not be measured in terms of reality, but rather in terms of practical utility. It is, therefore, important that characteristics which form utility are embedded into the approach. Lincoln and Guba [LG85] define several characteristics of quality, in this sense: credibility, the level of confidence in the findings; transferability, whether or not the findings have broad applicability; dependability, whether or not the findings are consistent with other venues and can be repeated; and confirmatory ability, the extent to which the findings are shaped by the phenomena under investigation and not bias. Historically, there have been many challenges to achieving quality in these areas within education research. The Tooley Report [TD98] warns of several pitfalls relevant to educational research which should be avoided: partisanship, a threat to confirmability in which authors become over-committed to a particular theory or otherwise fail to make a rigorous attempt at falsification; reliance on non-empirical sources, which can be appropriate for generating hypotheses but conclusions are subject to nearly all threats to quality; research focus, a threat to transferability, where the relevance is not to a pertinent problem in the field or the proposed solution has idiosyncratic impact; and lack of methodological rigour, a threat to the credibility of the research.

Rigour is of particularly concern in computing education research. Reviews of the literature have shown that, although quality is consistent in both conferences and journals [RJBS07, FCSC10], there is inadequate description in research reports and many projects contain methodological flaws [RJSL08b]. Given the prominent philosophies of the educators that the computing education research community aims to serve [Cle01], such weaknesses need to be addressed in order for their work to have impact.

3.3 A Pragmatic Design Approach

It is important to recognise that there is no single dominant paradigm within the physical and social sciences which adequately address all of the aspirations for computing education research. So, what will appropriate research look like? The following section describes an approach to *design research* for computing education research situated within the *pragmatic philosophy*.

3.3.1 Philosophical Underpinnings

3.3.1.1 Ontological Position

Ontology refers to ones beliefs about the nature of reality. Guba & Lincoln [GL+94] suggest there are three primary ontologies: critical realism, in which reality is independent of the knower and is assumed to exist but can only be imperfectly comprehended; relativism, in which reality exists as multiple intangible mental constructions which are socially and experientially based; and historical realism, in which reality is believed to be plastic and is shaped by a variety of socio-cultural forces and crystallised into what, for practical purposes, is a real immutable structure. The latter, of which, is the position taken in this thesis, although Herron & Reason [HR97, p. 258] extends the notion of this *transactional realism* by noting that "what can be known about the given cosmos is that it is always known as a subjectively articulated world, whose objectivity is relative to how it is shaped by the knower. But this is not all: its objectivity is also relative to how it is inter-subjectively shaped". Thus, a meta-reality independent of all observers is assumed to exist, which mediates the subjective conceptions of reality formed by observers, suggesting that multiple mediated realities are formed through a system of complex transactions between them (also see Dewey's work in this area [BB03]).

3.3.1.2 Epistemological Position

Epistemology refers to ones beliefs about the nature of knowledge and what constitutes acceptable contributions. Orlikowski & Baroudi [OB91] analyse the Information Systems literature and discuss three typical positions. These are: the positivist position, which is "premised on the existence of *a priori* fixed relationships within phenomena which are typically investigated with structured instrumentation"; the interpretativist position, which "assumes that people create and associate their own subjective and inter-subjective meanings as they interact with the world around them [...] attempting to understand phenomena through accessing the meanings that participants assign to them"; and the critical position, which "aim[s] to critique the status-quo through the exposure of what are believed to be deep-seated, structural contractions within social systems". Drawing on these three different positions, it is important to recognise that knowledge itself can be defined in different ways and what constitutes knowledge for one audience may not constitute knowledge for another. As such, the pragmatic epistemological position emphasises utility [BB03]. Hence, different forms of evidence are needed to demonstrate utility in different contexts and for different audiences.

This socio-cultural aspect to the conception of knowledge needs to be considered by education researchers. In particular, computing education researchers serve a community of computing educators whose dominant philosophies are grounded in the positivist tradition. Thus, the validity of observations is placed under key scrutiny and the hypo-deductive reasoning is privileged over the inductive and the abductive. It is the position of the author, therefore, to adopt a dual-epistemology in which evidence for utility for practitioners (e.g. the evaluation of constructs, models, methods, and instantiations) follow the positivist position (as defined by Orlikowski & Baroudi [OB91]), however a more pragmatic stance shifting between this position and the interpretive/critical positions is acceptable in the design stages where such knowledge is likely to only be shared within the computing education research community.

3.3.1.3 Methodological Position

Ones methodological position refers to ones beliefs about the nature of how knowledge should be constructed. The position taken in this thesis is *collaborative action inquiry*, as defined by Herron & Reason [HR97]. This states that knowledge construction is situated within a participative world-view in which critical subjectivity is enhanced by critical inter-subjectivity. Hence, the construction of knowledge is situated in practical activities and is constructed through a discourse between investigators, co-investigators, and practitioners. Within the framework of the ontological and epistemological position, this suggests that individual studies reveal knowledge both about our own respective realities as well as provide insight into our shared meta-reality and the key ways in which it mediated. Through discourse, insights are then provided into practice which improve the utility of our actions.

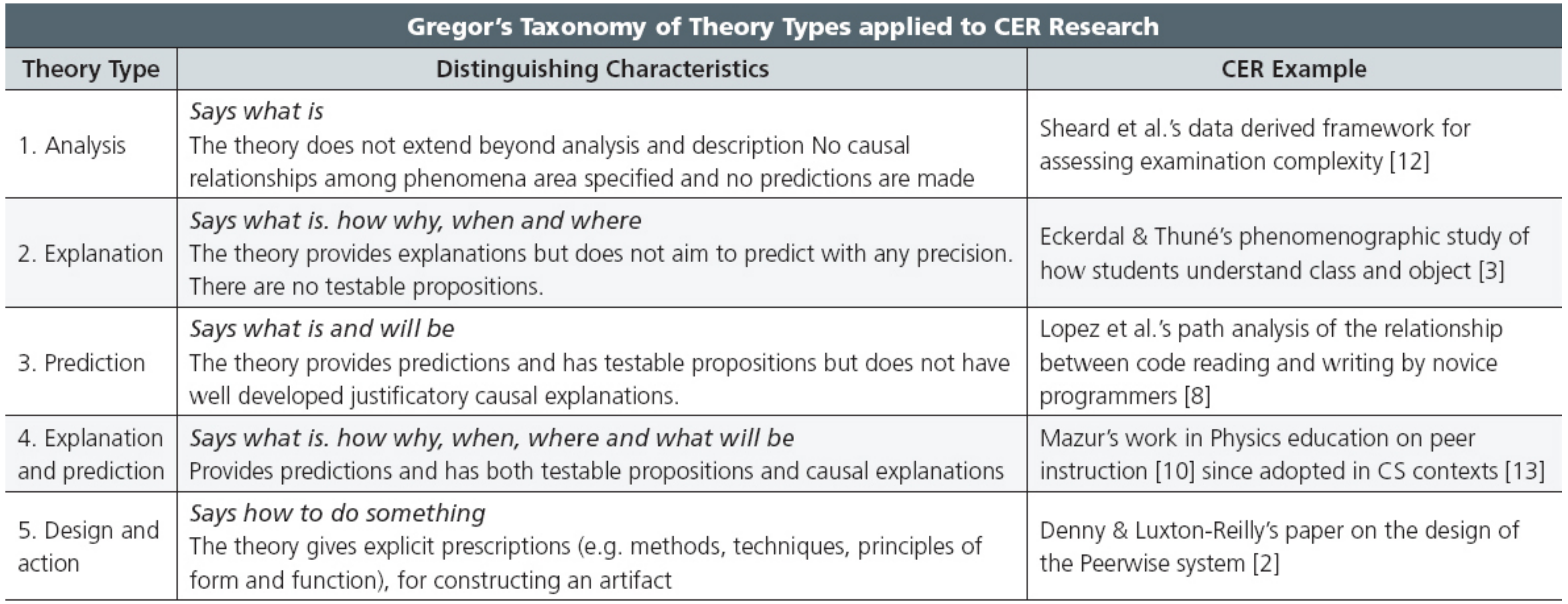
In line the ontological and epistemological framework, however, while advances in the design of artefacts can use a wide variety of approaches, there is a need to incorporate approaches that emphasise standardised measurement (in the form of quantitative data) and objectivity (e.g. survey and experimental methods) in their evaluation.

3.3.1.4 Axiological Position

The axiological position refers to what a researcher believes is intrinsically worthwhile. That is, what serves as the valued end. In an earlier section, the utility position was assumed such that it was stated that education research deals with the development of learning and the learning experience with the goal of enhancing and enriching educational practice. Herron & Reason put it as follows:

> The axiological question can also be put in terms of the ultimate purpose of human inquiry, since any ultimate purpose is an end-in-itself, a state of affairs that is intrinsically valuable. In the participative world-view the ontological account of reality as subjective-objective, as co-created with the given cosmos, leads over into the axiological question. For what purposes do we co-create reality? The answer to this is put quite simply by Fals-Borda: to change the world; or as Skolimowski points out, participation implies engagement which implies responsibility. The participative world-view is necessarily leads to an action orientation; not a impulsive action which, as Bateson describes it, cuts through the circuits of that natural world, but a reflective action, a praxis, grounded in our being in the world. [HR97, preprint, p. 4]

To expand upon this point, there are specific forms of research which have value in different ways. Clear illustrates, drawing on the work of Gregor [Gre06, Gre09], different types of theoretic contribution of, which is reproduced in Figure 3.2.



Gregor's Taxonomy of Theory Types applied to CER Research		
Theory Type	**Distinguishing Characteristics**	**CER Example**
1. Analysis	*Says what is* The theory does not extend beyond analysis and description No causal relationships among phenomena area specified and no predictions are made	Sheard et al.'s data derived framework for assessing examination complexity [12]
2. Explanation	*Says what is. how why, when and where* The theory provides explanations but does not aim to predict with any precision. There are no testable propositions.	Eckerdal & Thuné's phenomenographic study of how students understand class and object [3]
3. Prediction	*Says what is and will be* The theory provides predictions and has testable propositions but does not have well developed justificatory causal explanations.	Lopez et al.'s path analysis of the relationship between code reading and writing by novice programmers [8]
4. Explanation and prediction	*Says what is. how why, when, where and what will be* Provides predictions and has both testable propositions and causal explanations	Mazur's work in Physics education on peer instruction [10] since adopted in CS contexts [13]
5. Design and action	*Says how to do something* The theory gives explicit prescriptions (e.g. methods, techniques, principles of form and function), for constructing an artifact	Denny & Luxton-Reilly's paper on the design of the Peerwise system [2]

Figure 3.2: An Adaptation of Gregor's Taxonomy of Theory Types Applied to Computing Education Research from [Cle13, p. 29]

While not all forms of theory shown above will lead directly to practical utility for practitioners, this combination of approaches helps researchers understand more about learning and the learning experience, ultimately leading to such contributions in the future.

3.3.2 Research Strategy

3.3.2.1 The 'Twin Spiral' Model

The core strategy used in the studies presented in this thesis is design research (see [MR$^+$13] for details). It is fundamentally iterative and follows a two-stage design and evaluation approach with both being based on the empirical method. However, as outlined in the earlier section, there are different types of contribution which design research can make with respect to computing education field. Firstly, the understanding of constructs and models. Secondly, the application of methods and instantiations. As such, the research strategy constitutes two spirals, as shown in Figure 3.3 below.

The first spiral can be thought as being scholarship of discovery, where the focus is to improve constructs and models. The second spiral can be thought as being scholarship of applications, which is concerned with methods and instantiations when applied in practical contexts. The vortex produced, naturally, represents scholarship of integration. This enables contributions to pass between one scholar to another, or indeed, one contribution to another by the same scholar. As is common in pragmatic research, when a goal is achieved or a new goal is identified, the spirals can be pivoted in order to address the new goal. Thus,

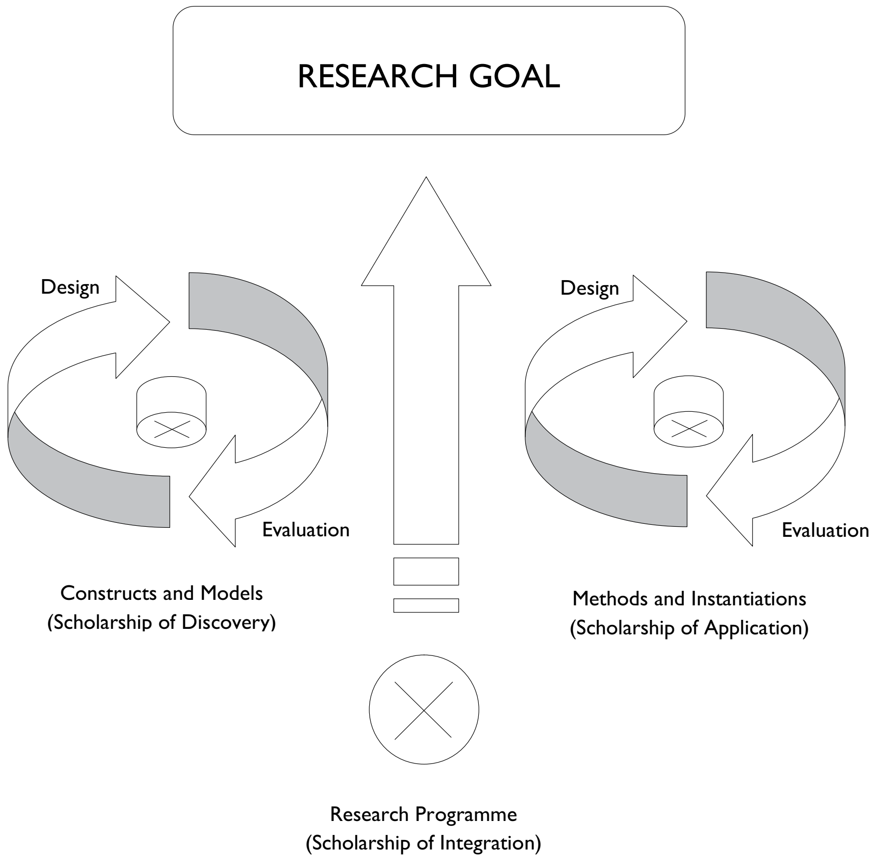

RESEARCH GOAL
Design
Evaluation
Constructs and Models
(Scholarship of Discovery)
Design
Evaluation
Methods and Instantiations
(Scholarship of Application)
Research Programme
(Scholarship of Integration)

Figure 3.3: The Twin Spiral Model of Design Research in Computing Education

several pivots are possible to re-align objectives based on findings made in any of the three parts of the strategy.

3.3.2.2 On the Choice of Methods

Within each spiral, a different research method can be used in each iteration of the spiral based on the findings of the previous iteration and the needs of the design. This is because different research methods serve different aims. They are tools to help researchers solve problems. Subsequently, the choice of method is based upon these different goals, for example: maximising generalizability with respect to populations; precision in control and measurement; and authenticity of the participants and context within which the research is conducted (also known as “existential realism”) [McG81]. This is shown in Figure 3.4.

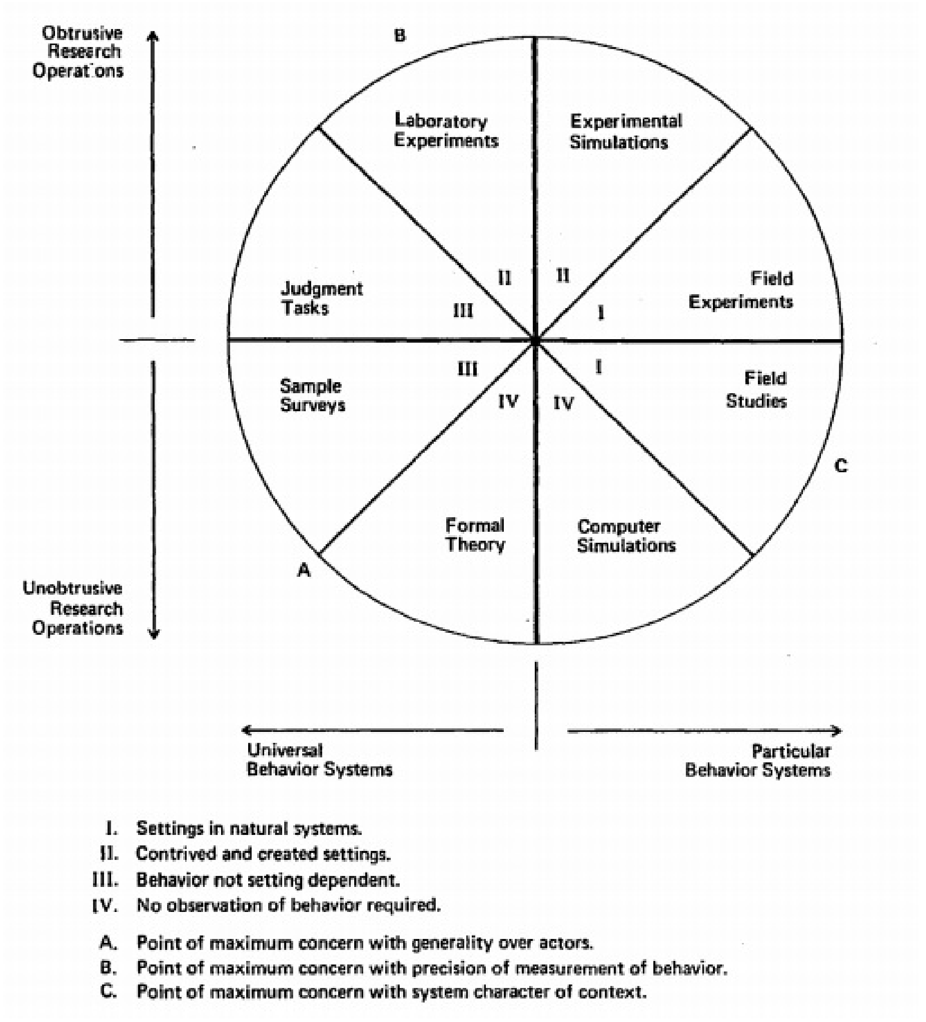

Obtrusive Research Operat ons
B
Laboratory Experiments
Experimental Simulations
Judgment Tasks
II
II
III
I
Field Experiments
Sample Surveys
III
IV
IV
I
Field Studies
C
Formal Theory
Computer Simulations
A
Unobtrusive Research Operations
Universal Behavior Systems
Particular Behavior Systems
I. Settings in natural systems.
II. Contrived and created settings.
III. Behavior not setting dependent.
IV. No observation of behavior required.
A. Point of maximum concern with generality over actors.
B. Point of maximum concern with precision of measurement of behavior.
C. Point of maximum concern with system character of context.

Figure 3.4: The Three Conflicing Desiderata in Dilemmatic Method Selection from [McG81, p. 183]

The maxima which can be achieved for these different goals are illustrated in Figure 3.4. McGrath [McG81] shows that by attempting to maximise one goal, the choice and operations made undermine the other two; and attempting to optimise two will minimise the third. Thus, every choice within the research strategy constitutes a “three horned dilemma” [p. 184] and research in this area must be plural and iterative in order to serve these different goals using different methods at different stages of the research project.

It can be seen in the table that some methods will more likely be used in the spiral that addresses the development of constructs and models (e.g. surveys), while others are more likely to be used in the development of instantiations (e.g. focus groups, experiments). Additionally, while less pronounced, different methods may be used in development stages and others may be used in evaluation stages.

3.3.3 On Validity

Clarifying Lincoln and Guba’s [LG85] notions of credibility, dependability, etc. is the concept of validity. This concept represents an assessment of the extent at which a claim can be considered sound in terms of logical reasoning and available evidence. It is a unitary concept [NS13] which exists upon a spectrum (i.e., validity can be strong or weak) but has many different aspects. Notably, the notions of *internal* and *external* validity introduced by Cook and Campbell [SCC02]. Internal validity refers to the design of a study, discerning whether or not claims are based on sound judgement. This implies that research methodologies need to endeavour to eliminate confounding factors and eliminate bias to ensure that sound reasoning can be derived from them. External validity, on the other hand, refers to issues of generalisability: can the findings from a particular study be assumed to apply to other similar contexts? Different research methods vary in terms of internal and external validity based on their design and the way participants are selected, respectively. It is generally accepted that the more probabilistic a design (e.g., random allocation) and a sampling method (e.g., random sampling) there is a lower possibility of bias and consequently the results are said to be more valid because the sample is more likely to representative.

3.3.4 On Ethics

Ethical approval will be sought for each work package from the Ethics Committee within the Department of Computer Science at Brunel University. The project will be conducted in line with ethical guidelines provided by the British Educational Research Association (BERA) and all relevant U.K. legislature.

3.3.5 Data Collection Techniques

Data collection is an important process in the research, irrespective of the choice of overall research strategy. A variety of methods can be used to collect qualitative and quantitative data, including observation, interview, focus group and questionnaire. Typically, in the pragmatic philosophy, a variety of data collection techniques are used from the broader range of two key types of data: quantitative or qualitative data. Notwithstanding the potential utility of qualitative data in the design of models and instantiations, the evaluation of models and instantiations often relies on quantitative data. The choice of quantitative data helps to provide standardised measurement of well-defined variables, enabling different designs to be compared and the relative size of differences between them to be calculated. Thus, providing some insight into the practical relevance and impact of any intervention.

In order to collect this data, questionnaires are used. This is because the use of Likert scales and Likert-type items are useful in measuring latent variables (i.e., not directly measurable) such as the attitudes, beliefs and emotions of participants [Lik32].

3.3.6 Data Analysis Techniques

The analysis of the work contained in this thesis focuses on quantitative data analyses which are based in the frequentist tradition. The choice of frequentist analysis techniques (e.g. t-test, χ^2 test, ANOVA, etc.) [GS90]. To this end, Predictive Analytics Software (PASW) v18 for Windows has been used to conduct these analyses throughout the research presented in this thesis.

3.3.7 Datasets Presented in the Thesis

The research described in the following chapters spans several years and as such several datasets were constructed using different samples. Many different measurement instruments were used, and in some cases participants completed multiple measurement instruments. Figure 3.5 illustrates the different samples collected, indicating the measurement instrument used as well as the times at which data collection was conducted. Different colours indicate different groups of participants.

Note that while some studies concluded in 2012 and 2013, the validation of measurement instruments did not conclude until 2014. As such, measurement validation was post-hoc.

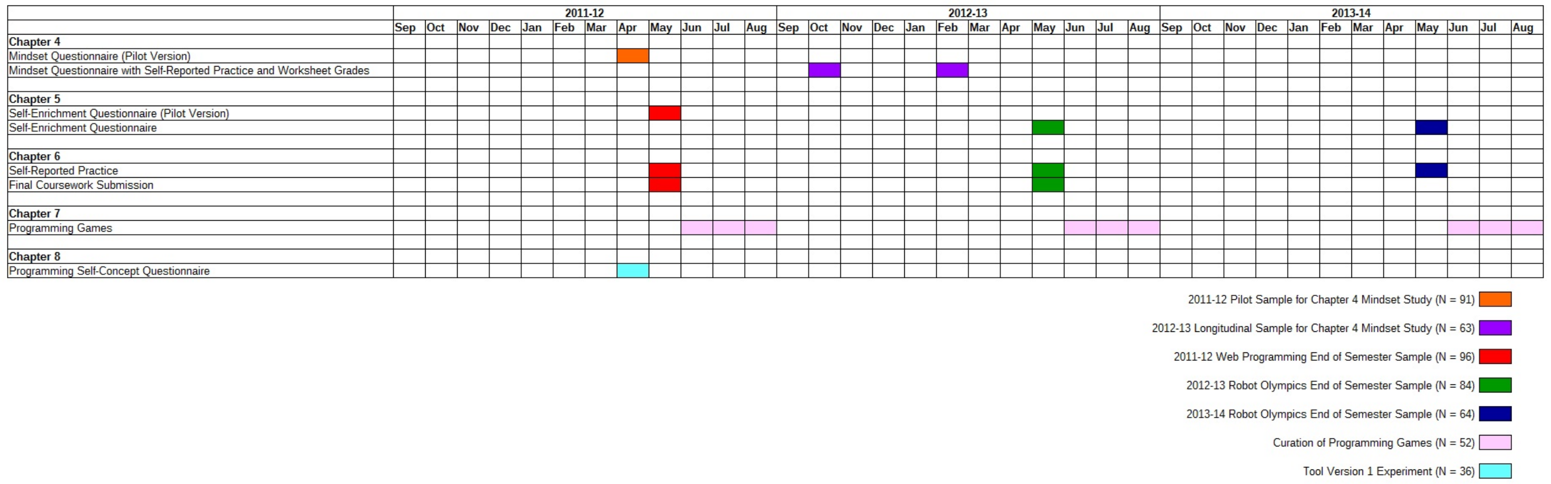


Figure 3.5: Datasets Presented in this Thesis

3.4 Summary

In summary, this chapter has discussed the role of scholarship in computing education and proposed an approach to research which aims to solve the problems outlined in Chapter 2. This is based on design research and the pragmatic philosophy, aiming to produce artefacts including constructs, models, methods, and instantiations which educators can use to enhance their practice, and thereby the learning and learning experience of their students.

4

On the Domain-Specificity of Mindsets

Research shows that those with different "mindsets" believe that either their traits can or cannot change. Focusing on mindset as a single construct, however, may not be appropriate in the computing context. This chapter presents findings from two surveys of undergraduate students, revealing that beliefs about intelligence and programming aptitude form two distinct mindsets. Additionally, the programming aptitude mindset has greater utility predicting programming practice. Consequently, these results suggest a domain-specific approach to future research on self-beliefs in computing education research.

4.1 Introduction

The conceptual model presented in Chapter 2 presumes a general model of beliefs which interact with emotions. However, it may be the case that a more domain-specific approach is needed because these beliefs will have a more domain-specific form. If so, such a form could likely have a more pertinent impact on the development of students' practice behaviour.

As such, this chapter will explore whether or not constructs such as mindset are domain-specific. The following section will introduce some additional background on mindsets and discuss how the theory has been used in the computing education research field. The next sections will then describe the research questions and the hypotheses. This leads into a description of two studies which consisted of a round of questionnaires in 2011-12 and two further rounds of questionnaires in 2012-13. The chapter then closes with a brief overview of the findings and a discussion on their implications for furture research.

4.2 Background

Educators often situate high levels of scaffolding and formative feedback within the introductory programming lab [Jen01]. Despite such efforts, however, many beginners do not practice regularly. Often, such students claim they experience apprehension and discomfort when they attempt to do so [RS10b]. These emotional responses can prompt students to stop working on difficult assignments [KS10b]; it has been reported that such affective factors worsen over a course of programming instruction [MD04]. However, not all students react this way when they encounter problems. For example, some perceive compilation errors as a challenge to be overcome rather than as an indication of failure [MT08]. A potential reason for such conflicting perspectives is that students have different ways of reflecting upon their learning [KS10b, KS12]. These differences may correspond to students' self-beliefs [MT08], presenting an opportunity for educators to nurture particular mindsets.

According to the self-theories proposed by Dweck [Dwe99], individuals hold beliefs about the nature of their personal traits, referred to as their mindset, which can be classified into one of two core beliefs. Those with the fixed mindset believe their traits are an entity that cannot be changed. Conversely, those with the growth mindset believe their traits are flexible and can be enhanced through effort.

These beliefs have implications for the way that students engage in self-regulated learning. This is because the learning strategies that students apply depend on whether they believe such strategies are necessary for learning, and are effective at addressing problems [DM08]. As a result, those with a fixed mindset tend to adopt a helpless response when they encounter difficulty. In contrast, those with a growth mindset tend to persevere, adopting a mastery-orientated strategy [DD78, YD12].

In order to nurture a growth mindset, educators embed growth messages such as "the brain is like a muscle, it develops through exercise" into their teaching practice. However, this advice is often framed in terms of intelligence [MT08, KS12, SHCD09]. Do students generalize such messages? The human mind can be conceived in terms of multiple intelligences [Gar06], and self-theories have been adapted for areas as varied as shyness [Bee02], maths-ability [GRD12] and willpower [JDW10]. Therefore, it is conceivable that students do not associate their programming ability with a general sense of intelligence, but rather to a sense of programming aptitude.

Programming has been described as a discipline that presents radical novelties to beginners [Dij89]. This is because new students often need to adapt their way of thinking to accommodate the abstract and intangible concepts that are applied in program creation [CB07], as such thinking is seldom developed prior to their first programming course [Jen02]. Hence, the discipline can feel distinct, potentially promoting a separate mindset for programming aptitude.

This has implications for the design and evaluation of teaching practice. A "saying is believing" exercise required students to "describe a time when (they) learned something other than programming (...) but with practice and perseverance (they) were able to succeed" [SHM+08b, p. 176]. However, this was shown to lack practical impact. This may be because students can hold a separate mindset for programming, in which case reflecting on past success in another discipline may not succeed.

Another intervention attempted a rich combination of mindset-informed training and feedback practices [CCD+10]. This was shown to have some success, but only for students who also received a programming-specific crib-sheet which contextually reinforced the growth belief. This could indicate an advantage in applying educational practices that are designed to change a more specific mindset, as opposed to a general mindset for intelligence [c.f. [FVC09]]. However, only a single measure was used in the study.

4.3 Research Questions

Following this line of reasoning, a mindset scale can be adapted to form two distinct sub scales: items about programming aptitude and items about general intelligence. However, the constructs may be correlated, which raises several research questions (RQs) about the implications of separate mindsets for teaching practice in the introductory programming context:

RQ1. Is mindset for programming aptitude empirically distinct from mindset for general intelligence?

RQ2. Can students have a mindset for programming aptitude that is substantially different to their mindset for general intelligence?

RQ3. Does the mindset for programming aptitude have more utility for predicting programming practice compared to the mindset for general intelligence?

RQ4. Does the mindset for programming aptitude change differently to the mindset for general intelligence over a period of programming instruction?

4.4 Hypotheses

To explore the relative merits of modelling separate mindsets, there is a need to establish the differences between them.

The first research question examines one hypothesis relating to the factor structure of the measurement instrument. It is predicted that a confirmatory factor analysis will show that a two-factor model will not be significantly different to the observed factor structure (H1).

The second research question examines two hypotheses: firstly, that a model with programming aptitude mindset and intelligence mindset as two distinct, but slightly correlated factors (H2); secondly, for the notion of separate mindsets to have utility for educators, the classification of each mindset (being either fixed or growth) should not have a high level of consistency (H3).

The third research question then explores the impact of each mindset on programming practice behaviour. It is hypothesized that both programming aptitude mindset and mindset for intelligence (H4 and H5) are related to programming practice behaviour. Given their relation to resilience [DD78, YD12], each relationship will be moderated by early performance (H6 and H7), such that those achieving high grades will not be as strongly influenced by their mindset. However, each will have a different level of explanatory power on programming practice behaviour (H8).

The fourth research question investigates change in mindset over time. As there could be elements of programming instruction that induce a fixed mindset [MT08, CCD$^+$10], it is hypothesized that mindset for programming aptitude will become more fixed over a period of programming instruction (H9). Mindset for intelligence may also change (H10), but less so than programming aptitude (H11).

4.5 Method

4.5.1 Data Collection

A pilot survey addressing RQ1 was conducted in 2011-12 to examine these hypotheses. Participants were recruited from two core programming modules at the authors' institution. The study was promoted via pre-registered email, institutional email, notices on BlackBoard Learn, and through a course-related Facebook Group. The survey was also distributed in the final laboratory session of the year in April 2012.

A two-wave survey addressing RQs 1-4 was conducted in 2012-13 to examine these hypotheses. Participants were recruited from two core programming modules at the authors' institution. The study was promoted via pre-registered email, institutional email, notices on BlackBoard Learn, and through a course-related Facebook Group.

The questionnaires were distributed using SurveyMonkey, and were available for 11 days across the 8th and 16th week of the semester, respectively. Participation was voluntary. In order to identify programming assessments corresponding to each respondent, student identification numbers were either concealed and encoded into hyperlinks, or reported. The sampling frame consisted of 296 first- and second-year undergraduate students on programming modules within the authors' institution. To be eligible, students had to be at least 18 years of age and had to have submitted their first three lab assignments, the deadlines for which were prior to the date the survey was conducted. In the first wave of the survey 73 students completed all of the items, giving an initial response rate of 24%. However, there was some attrition between the first and second wave of the survey, with only 63 students responding to both. Thus, the attrition rate was 14%.

4.5.2 Participants

Participants were first and second year undergraduate students following the sequential pathway for "Computer Science" or "Business Computing". The descriptive statistics for the second sample show that approximately 24.3% of the respondents were female, and that the average age was 20 years ($\bar{x} = 20.48, \sigma = 2.42, max = 30$), with 17.6% of respondents being over the age of 23 at entry.

As the response rate was low and the early programming scores for the sample indicated that many were performing at a high merit level ($\bar{x} = 6.61, \sigma = 1.71, max = 9.00$), there was concern about response bias. However, performance did not significantly differ across the cohort ($\bar{x} = 6.35, \sigma = 1.58, t[72] = 1.291, p = .201$). Furthermore, the proportion of mature ($age > 23$) ($\chi^2 = 2.647, p = .103$) and female students ($\chi^2 = 1.372, p = .241$) was typical of the cohort.

Admission to the pathway required at least 300 UCAS Points (University & College Admission System Points), with a strong preference for STEM subjects (science, technology, engineering, and mathematics). Prior programming experience was not required (44.6%). However, students without a relevant STEM qualification, or the required points, could opt to pursue a relevant foundation course (9.6%).

4.6 Measurement

The questionnaire measured three latent variables: mindset for intelligence (INTEL); mindset for programming aptitude (APT); and programming practice behavior (PRACT). Common factor analysis techniques were used to generate these scores, rather than principle components analysis, to explore how the underlying structure of items shared variance reflected the latent variables of interest (see [BLR+13] for a discussion). Early programming performance (GRADE) was measured using the first three assignments in each module.

4.6.1 Mindset For Intelligence

To measure mindset for intelligence, items were drawn from Dweck's mindset scale [Dwe99]. Five items were used, including three statements that endorsed the fixed belief (such as "my intelligence is something about me that I can't change very much") and two statements that endorsed the growth belief (such as "I can always substantially change how intelligent I am"). These were presented as a 7-point Likert scale, ranging from strongly disagree to strongly agree. The order in which items were displayed was randomized alongside items measuring mindset for programming aptitude. Composite scores were generated using a regression method based on the factor score matrix generated by a maximum-likelihood analysis. A high score indicated a fixed belief.

4.6.2 Mindset For Programming Aptitude

To measure mindset for programming aptitude, items were again drawn from Dweck's mindset scale [Dwe99], but adapted to the programming context. Five items were used: three of these statements endorsed the fixed belief, for example: I have a fixed level of programming aptitude, and not much can be done to change it; the remaining two items endorsed the growth belief, for example: I believe I am able to achieve a high level of programming aptitude, with enough practice.

The items were presented as a 7-point Likert scale, with responses ranging from strongly disagree to strongly agree. The items were presented randomly, alongside items measuring mindset for intelligence. Composite scores were generated using a regression method based on a factor score matrix produced by a maximum-likelihood analysis. A high score on this scale indicates a fixed belief.

4.6.3 Regularity of Programming Practice

A measure of programming practice was created for this survey using a 7-point and a 4-point item. These were presented as Guttman-type items, providing an indication of frequency of practice and the typical duration of each practice session. The questions were: in a typical week of study I find myself writing code during the closed-labs / at least 1-5 day(s) a week / every day and in a typical session I concentrate on programming for up to 30 minutes / at least 30 minutes / at least one hour / at least two hours. Responses were compiled into a single composite score using principal axis factoring. As this was a retrospective self-report measure, potential self-report biases mean it should be interpreted with caution and not treated as actual practice [DGV02a].

4.6.4 Early Programming Performance

As the core programming modules used the same assessment structure, early programming performance was measured using existing assessment data. Assignments were assessed as code reviews by a team of Ph.D. students covering the modules, who typically had good consistency ($ICC = 0.73$) based on six submissions. Grades reflected the functional coherence of solutions, the presence of common pitfalls, and a judgement on quality according to a rubric. They were recorded as 1 (pass), 2 (merit), and 3 (distinction). The results of the first three assessments were added together to form a composite score.

4.7 Data Analysis

The data was analyzed using PASW 18.0.3 and AMOS 21.0.0 for Windows. All cases were included. Cases with missing data were removed list-wise. All reported p-values are two-tailed with significance determined at the .05 level. As multiple hypotheses were explored, p-values were adjusted to control for the false discovery rate ($FDR = .05$) using the Benjamini-Hotchberg Procedure [BH95]. Note, H7 and H10 are not associated with a null hypothesis significance test.

4.7.1 The Two-Mindsets Factor Structure

The data collected for the pilot study was analysed using the generalized least squares method of confirmatory factor analysis in AMOS 21.0.0, with all cases included in the analysis. The results are shown in Table 4.1.

Table 4.1: Fit Indices and Criteria for Both Models (Pilot Study)

Fit Index	1-Factor Model	2-Factor Mode	Adequete Fit Criteria [HBBA10]
SRMR	.146	.077	$< .08$
CFI	.507	.959	$> .90$
RMSEA	.092	.027	$< .08$
Bollen-Stein p	.062	.369	$> .05$

Note: SRMR: Standardized Root Mean Square Residual; CFI: Comparative Fit Index; RMSEA: Root Mean Square Error of Approximation; N = 94.

The results show that the single factor model, based on the notion of a single dominant self-theory, was significantly different to the data ($\chi^2 = 64.330, df = 36, p = .003$). In contrast, the two-factor model, where intelligence and programming aptitude are distinct self-traits, had adequate fit ($\chi^2 = 36.382, df = 34, p = .358$). This is further illustrated by the fit indices presented in Table 4.1. This suggests that conceptions towards different self-traits should be considered separately when extending self-theories to specific domains, such as programming education.

As with the pilot study, the data collected in the second survey was analysed using confirmatory factor analysis. However, in this case as no heteroskedasticity was found and the likelihood of a Haywood-case was low, the maximum-likelihood method was used. The results are summarized in Table 4.2:

Table 4.2: Fit Indices and Criteria for Both Models (Actual Study)

Fit Index	1-Factor Model	2-Factor Mode	Adequete Fit Criteria [HBBA10]
SRMR	.108	.078	$< .08$
CFI	.728	.928	$> .90$
RMSEA	.117	.061	$< .08$
Bollen-Stein p	.015	.313	$> .05$

Note: SRMR: Standardized Root Mean Square Residual; CFI: Comparative Fit Index; RMSEA: Root Mean Square Error of Approximation; N = 73.

The results show that a two-factor model had better fit ($\chi^2 = 43.094, df = 34, p = .136$) than a single-factor model ($\chi^2 = 69.619, df = 35, p = .000$). Furthermore, the items used to measure mindset for intelligence ($\alpha = .73$) and mindset for programming aptitude ($\alpha = .61$) demonstrated marginal, but adequate reliability.

4.7.2 Consistency Between Different Mindsets

Participants were classified using a two-step clustering procedure that was applied separately to APT and INTEL. Each showed the expected two-cluster solutions, based on the log-likelihood distance and the Bayesian information criterion. The average silhouette coefficient was used to evaluate the clustering solutions, yielding values greater than 0.7 for both analyses indicating that the solutions were a 'good' fit. The classification of each individual student is shown in Table 4.3:

Table 4.3: Mindset Classifications for Individual Students

	Fixed Intelligence	Growth Intelligence	κ	p
Fixed Programming Aptitude	10	13		
Growth Programming Aptitude	11	39	.220	.060

The correlation between factor scores was significant ($r = .248, p = .034$). However, the level of agreement between each classification scheme, based on the kappa statistic, indicated only *fair* agreement ($p = .060$) [LK77]. It can be seen that 23 students were classified as fixed APT (31.5%), while 21 students were fixed INTEL (28.7%). Inconsistency occurred in 24 cases (32.9%), where students held different beliefs for the two domains. This was most prominent for those with fixed APT, where 13 cases maintained growth INTEL (56.5%). However, 11 cases with a growth APT also had inconsistent beliefs (28.8%).

4.7.3 Impact of Each Mindset on Practice Behaviour

Two linear regression analyses compared the independent impact of APT and INTEL on PRACT. Assumptions of residual normality, independence, and homoskedasticity were verified prior to each analysis. The model exploring APT was significant ($p = .003$) and is described below in Table 4.4.

Table 4.4: Programming Aptitude Mindset Regression Model

Construct	β	$\sigma_{\bar{x}}$	t	p
Programming Aptitude Mindset (APT)	-.249	.109	-2.225	.025
APT * GRADE	.245	.101	.2.254	.027
Early Programming Performance	.248	.108	.2.281	.026

Note: Adjusted R^2 *= .141; N = 73;* $F[3, 70] = 4.985, p = .003$.

The relationship, illustrated in Figure 4.1, reveals that those with fixed APT and low GRADE tended to practice less than their peers. However, students with high GRADE were not as strongly influenced by their APT. This interaction, however, was not found in the model exploring INTEL, shown in Table 4.5.

Table 4.5: Intelligence Mindset Regression Model

Construct	β	$\sigma_{\bar{x}}$	t	p
Intelligence Mindset (INTEL)	-.253	.114	-2.222	.030
INTEL * GRADE	.135	.107	.1.194	.237
Early Programming Performance	.283	.113	.2.490	.015

Note: Adjusted R^2 *= .089; N = 73;* $F[3, 70] = 3.385, p = .023$.

The expected interaction with GRADE was not significant ($p = .237$). Nevertheless, using INTEL to predict PRACT was significant ($p = .023$). Thus, both models had utility for predicting PRACT. However, the comparison shown below in Table 4.6 reveals several differences.

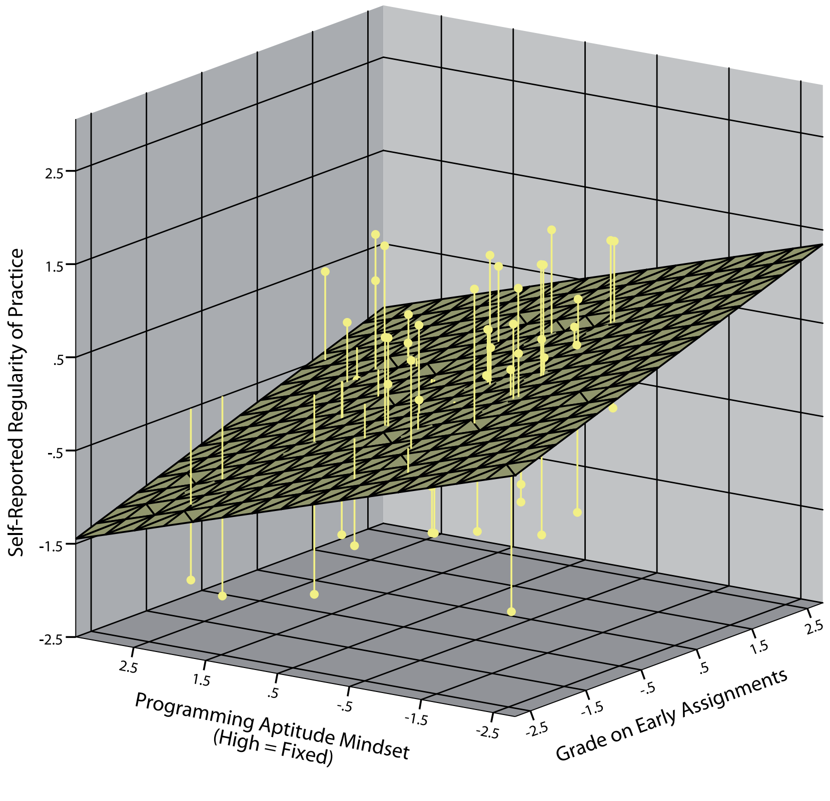


Figure 4.1: A 3D Scatter Plot Illustrating the Influence of Programming Aptitude Mindset and Performance on Early Programming Assignments on Self-Reported Programming Practice.

It can be seen that the regression model using the mindset scores for programming aptitude explained a larger proportion of variance ($Adjusted\ R^2 = .141, \Delta R^2 = .052$). Furthermore, there was a noticeable improvement in fit ($\Delta AIC = 4.246, \Delta AIC > 2$ [BA02]). Thus, the data shows that the APT model has greater utility for predicting PRACT.

Table 4.6: Model Selection

Model	*Adjusted* R^2	*AIC*	*PC*	*BIC*
Programming Aptitude Mindset Model	.141	-9.037	0.895	-2.166
Intelligence Mindset Model	.089	-4.791	0.948	2.081

Note: AIC: Alkaike Information Criterion; PC: Prediction Criterion; BIC: Bayesian Information Criterion.

4.7.4 Change in Belief for Each Mindset Over Time

A series of paired t-tests examined whether students' mindsets had changed between the first wave of the survey and the second wave. The results are shown in Table 4.7:

Table 4.7: Model Selection

Mindsetl	$\bar{x}_{w=0}$	$\bar{x}_{w=1}$	$\bar{x}_{\Delta}$	σ_{Δ}	t	p	d
Intelligence	-0.15	-0.09	.059	.508	0.927	.358	n.s.
Programming Aptitude	-1.16	-0.90	.267	.856	2.475	.016	0.62

Note: N=63; df = 62; w: survey wave (8 weeks between each wave); d: Cohen's d effect size.

There was a non-significant decrease in mean INTEL score ($p = .358$). Thus, students' INTEL remained stable across the eight-week period. However, there was a significant increase in APT score ($p = .016$). This suggests that students beliefs towards programming aptitude had become more fixed, with medium effect ($d = 0.62$) [Coh92]. In context, however, the mean difference ($\bar{x}\Delta = .267$) does not represent a large shift for the entire cohort. Only 30.2% of the respondents came to believe more strongly in a fixed perspective, with only 18% of cases changing distinctly from the growth belief.

4.8 Discussion

The literature on self-beliefs and motivation shows that mindsets can influence resilience [DD78, YD12]. Students with growth beliefs tend to continue to practice when they encounter difficulty. Those with fixed beliefs do not. Thus, it is important that educators inspire growth beliefs, as ongoing practice is important for developing expertise [EKTR93, Win96]. However, mindset may not reflect a single general construct focused on intelligence. This chapter shows some evidence that students may develop domain-specific beliefs in the area of computer programming.

The first research question examined whether students' beliefs about their intelligence and their beliefs about their programming aptitude could be substantially different. Although a significant correlation was found, the classification schemes showed low levels of agreement. Most students with the fixed belief for programming aptitude had the growth belief for intelligence. This suggests that students can have markedly different mindsets across domains.

The second research question explored the relationships between each mindset and programming practice behaviour. Although the results were modest, the regression model based on aptitude beliefs had a closer fit to the data and explained a greater proportion of the variance. Furthermore, while early performance in programming moderated the relationship between practice and aptitude beliefs, this was not found in intelligence model. These results seem to reinforce the notion that students do not associate their performance in computer programming with their sense of intelligence and suggest that the mindset for programming aptitude could have greater utility for predicting programming practice.

The third research question asked whether beliefs changed across an eight-week period of instruction. Although beliefs about intelligence did not change, it is a concern that nearly one-third of respondents came to believe more strongly in the fixed perspective of programming aptitude. The cause, in this case, is unclear. The literature suggests that many aspects of programming instruction [MT08, CCD+10] and feedback style [MD98, RGD12] could have contributed to the change. However, there could be differences in source as well as sensitivity. Thus, factors that affect domain-specific beliefs should be further explored.

4.9 Limitations

The study presented in this chapter has several limitations; notably, threats to external validity as students were recruited from two classes at a single institution and the number of students encountering early difficulties was low. Furthermore, the sample size constrained statistical power, so interaction effects could not be investigated. It should also be noted that the reliability of the mindset measure was marginally adequate, suggesting a need for further scale development (see [TD13]). In addition, it should be noted that the participants of the pilot study (2011-12) were exposed to teaching methods centred on exams and web programming while those in the main study (2012-13) were exposed to teaching methods centred on worksheet-orientated code reviews and robot programming.

Table 4.8: Summary of Findings and Adjusted P-Values

RQ	H_n	Hypothesis	$\tilde{p}$	Conclusion
1	H_1	$\chi^2 = 0$	.136	*Supported*
2	H_2	$APT \leftrightarrow INTEL$	.048	—
	H_3	$\kappa(APT \leftrightarrow INTEL) \neq 0$	.075	*Supported*
3	H_4	$APT \rightarrow PRACT$	.048	*Supported*
	H_5	$INTEL \rightarrow PRACT$	.048	*Supported*
	H_6	$(APT * GRADE) \rightarrow PRACT$	.048	*Supported*
	H_7	$(INTEL * GRADE) \rightarrow PRACT$	.263	—
	H_8	$\lvert AIC(APT) - AIC(INTEL)\rvert > 2$	—	*Supported*
4	H_9	$\bar{x}_\Delta(APT) \neq 0$	.048	*Supported*
	H_{10}	$\bar{x}_\Delta(INTEL) \neq 0$	.358	—
	H_{11}	$\lvert d(APT) - d(INTEL)\rvert > 0.2$	—	*Supported*

Note: Univariate tests for gender have been excluded as no multivariate significance was found in H5; $\tilde{p}$: Benjimini-Hochberg adjusted p-value; PRACT: Self-Reported Hours of Programming Practice Per Week; QLTY: Overall Code Quality; BOT: Enrolment in Robot-Centred Course; GEN: Gender; FC: Functional Coherence of Code; RD: Readability of Code; SO: Sophistication of Code.

4.10 Summary

As shown in Table 4.8, this chapter reveals some evidence that mindset for programming aptitude is not only distinct from mindset about intelligence, but that it may also have a stronger relationship with programming practice. This suggests a discipline-specific perspective may be appropriate when extending self-theory research into the software engineering context. As such, educators should emphasize the malleability of programming skill directly by, for example, contextually situating growth messages within relevant programming materials (e.g., code review rubrics [CCD+10]). Moreover, future work should examine measures of programming aptitude mindset and further investigate mindset interventions.

Declaration

Some of the work presented in this chapter can also be found in the following publications:

Scott, M. J. and Ghinea, G., Implicit Theories of Programming Aptitude as a Barrier to Learning to Code: Are They Distinct from Intelligence?, *Proceedings of the 18th ACM Annual Conference on Innovation and Technology in Computer Science Education (Kent, UK)*, July 2013, p. 347

Scott, M. J. and Ghinea, G., On the Domain-Specificity of Mindsets: The Relationship Between Aptitude Beliefs and Programming Practice, *IEEE Transactions on Education*, 57 (2014), pp. 169–174, DOI: 10.1109/TE.2013.2288700

5

Measuring Enrichment: Assessing Self-Beliefs in CS1

The previous chapter shows that domain-specific approaches to computing education research are warrented. However, a broad range of constructs are likely to be important in such work. It is therefore important that these constructs can be measured in a valid and reliable way. To this end, this chapter proposes a new research instrument based on four constructs: programming self-concept; interest in software development; programming anxiety; and mindset towards programming aptitude. Analysis reveals that the proposed instrument possesses adequete psychometric properties for use in future research.

5.1 Introduction

Reliable and valid measurement is critically important in educational research. Variables of interest should be clearly defined and measured with minimal error in order to make meaningful conclusions from an analysis [DeV12, TD13].

However, there is not a strong history of instrument development and validation in computer science education research. Two systematic reviews of the literature found that the quantitative research in the field would benefit from improvements in methodology and reporting [RJSL08a, Val04]. Of particular interest, only 1.5% of articles published between 2000 and 2005 reported adequate psychometric information to support the validity, the reliability, and the generalisability of their claims [RJSL08a].

A number of measurement instruments have since been developed and evaluated. For example, the Foundations of CS1 Test (FCS1) assesses students' performance in the cognitive-domain of introductory computing [TG11]. However, few instruments address the affective-domain of learning computing (e.g., attitude development [DT13]). In particular, there is a need to explore the emotive aspects of learning computer programming as, for some students, programming invokes strong negative feelings [Hug04, KS10a, RS10a] and shapes their self-beliefs in counter-intuitive ways [KB12]. This is important to consider because self-beliefs play an important role in academic development [BTD07, KBM73]. As an example, the previous chapter showed that beliefs about the nature of programming aptitude, extending Dweck's mindset theory (see [CCD+10, Dwe99, DM08, MT08, SHM+08a]), can lead to significant differences in the time that students report practising programming. However, to pursue this line of research further, a valid measurement instrument is needed.

As such, this chapter will propose a measurement instrument and will then address the research question: is the proposed measurement instrument reliable and valid? The following section will develop the conceptual model in light of the findings in Chapter 4, a parsimonious set of key variables is then identified to include in the measurement instrument. The next section then describes how the proposed measurement instrument was assembled. This leads into an evaluation with three cohorts of undergraduate computer students. The chapter then closes with a brief discussion of the potential uses of the measurement instrument, its limitations, and a conclusion on its adequacy for future research.

5.2 Proposed Conceptual Framework and Instrument Assembly

To validate the framework and test such a hypothesis, it is necessary to develop an appropriate measurement instrument. Therefore, in line with the proposed conceptual framework shown in Figure 5.1, items for the key variables were assembled.

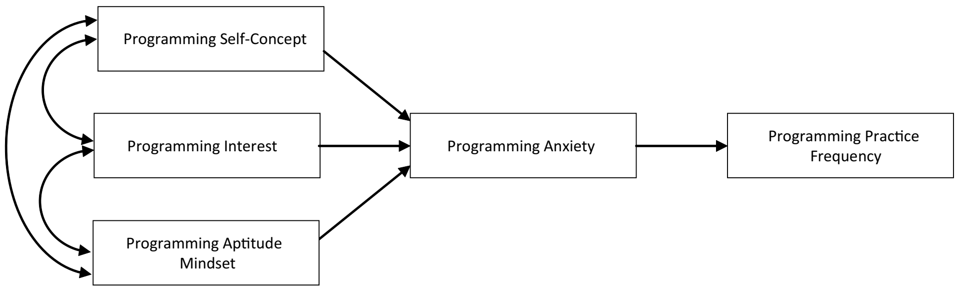


Figure 5.1: A Conceptual Framework for Enhancing Students' Programming Practice

The measurement model for the proposed measurement instrument consisted of four constructs: Programmer Self-Concept (PSC); Interest in Software Development (INT); Programming Anxiety (ANX); and Mindset Towards Programming Aptitude (APT). Additionally, in order to ensure appropriate discriminatory power between constructs, such as differences between self-concept and self-efficacy [BS03], items relating to software debugging task self-efficacy are also included (DSE). As existing instruments target similar constructs of interest, items were drawn from the literature and adapted to the introductory programming context. A self-report of programming practice behaviour, for the purpose of establishing the concurrent validity of the proposed framework, is also included.

The construct debugging task self-efficacy captures learners' cognitive self-assessments of whether or not they are confident in their ability to write and debug simple programs. This is based on the theoretical construct proposed by Bandura [Ban77], as it relates to how self-assessments influence behaviour change. The items for this construct were created using guidelines regarding the domain-specificity of self-efficacy and its association with particular criterial tasks.

The construct of programmer self-concept has some conceptual overlap with debugging self-efficacy, however there are a range of theoretical and empirical differences [BS03, FVC09]. It represents a composite of self-perceptions that one can be a good programmer, which is "formed through experience with and interpretations of one's environment". This construct drives the affective elements of being a programmer as opposed to a cognitive assessment of success at programming because "self-concept better predicts affective reactions such as anxiety, satisfaction, and self-esteem, whereas self-efficacy better predicts cognitive processes and actual performance" [BS03]. The items for this construct were adapted from scales used by Ferla et al. [FVC09] and Eccles & Wigfield [EW95]. These focus on the ability-belief component of self-concept.

The construct for interest in software development measures the extent to which an individual enjoys engaging with programming-related activities. This construct is believed to have a reciprocal relationship with self-concept, resulting in the pursuit of more achievement experiences in a domain [GMB03]. The items for this construct were adapted from the scale used by Wigfield et al. [WEY$^+$97], focusing on the enjoyment aspect of interest.

The programming anxiety instrument construct measures the self-reflected state of experiencing negative emotions, such as nervousness or helplessness, while writing and debugging programs. The items were drawn and adapted from the worry-component of the instrument used by Wigfield and Meece [WML88].

The mindset towards programming aptitude instrument construct represents the strength of a learners' belief in the notion of a fixed programming aptitude (e.g., aptitude is inherent and cannot change). The items were drawn from Dweck [Dwe99].

These items were then put together as a 5-point Likert instrument, with each item rated from strongly disagree to strongly agree. Each item was reviewed by 2 colleagues and a small convenience sample of undergraduate students, and revised to improve content validity and readability. This resulted in the instrument shown in Table 5.1 on page 76.

5.3 Research Questions

Having assembled the instrument, the following research question can be addressed:

RQ5. Does the proposed measurement instrument demonstrate adequete validity?

5.4 Hypotheses

In order to evaluate the proposed measurement instrument, its psychometric properties must be examined. Namely, based on the recommendations of Straub *et al.* [SBG04] and other authors (e.g. [DeV12, TD13]), reliability and validity need to be established in order to deem a measurement instrument adequate. However, what does "validity" mean?

Newton and Shaw [NS13] present a history of validity and its use in the psychological literature. They reveal that, despite appearences, there are a number of challenges associated with the concept. Notably, that although there are many ways of evaluating validity using different forms of evidence, many authors agree that it is a unitary concept and should be treated as such. That is, each *aspect* of validity is equally important. Addtitionally, they argue that validity is based on subjective degrees of confidence rather than absolutes. As such, the threshold for deciding whether or not an instrument is valid depends on the context of its use. For example, in the context of measure reliabilty (using Cronbach α), Nunally poses that the "satisfactory level of reliability is depends on how a measure is being used. In the early stages of research [...] one saves time and energy by working with instruments that have only modest reliability, for which purpose reliabilities of .70 or higher will suffice [...] In contrast to the standards in basic research, in many applied settings a reliability of .80 is not nearly high enough" [NBB67, p. 245-246].

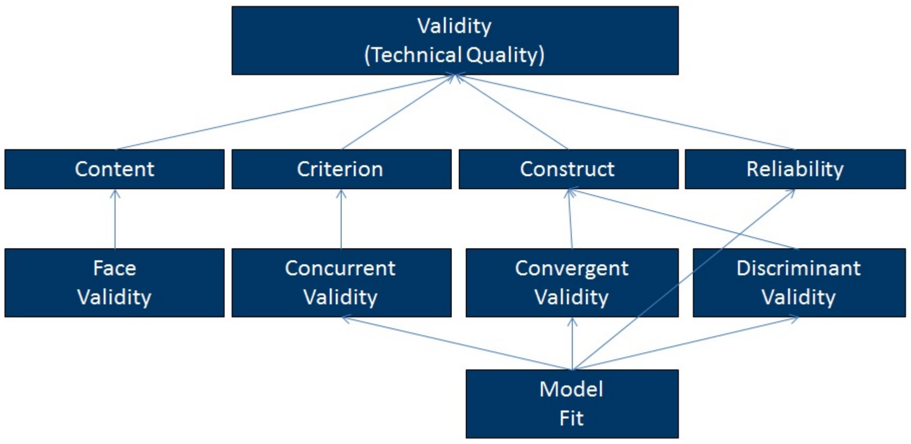


Figure 5.2: Aspects of Validity Drawn From [SBG04, HBBA10, NS13]

With this in mind, what forms of validity are important to make a decision about overall validity? According to Straub *et al.* [SBG04], there are three main forms of validity and reliability which are important in instrument development. These are shown in Figure 5.2. Content validity is the level at which items used to measure a construct reflect the meaning of the construct (and breadth of possible items which could represent the construct) to which the items will be generalised. Construct validity is the form of validity that deals with the degree to which items are an effective measure of a theoretical construct. This is often sub-divided into convergent validity and discriminant validity as evidence for both *imply* construct validity [HBBA10]. Convergent validity refers to the level at which multiple items which theoretically should be related are actually related. Conversely, discriminant validity assess the extent to which items which should be unrelated are actually unrelated. Reliability refers to the extent to which parallel items are consistent in what they are intended to measure (e.g. responses to a set of related items are internally consistent). Concurrent validity is also a consideration in cases where constructs should be related. That is, a construct is related to, or able to predict, another in the same instrument. As such, there are four hypotheses, relating to: content validity (H1); overall statistical model fit (H2); reliability (H3); construct validity in the form of convergence (H4) and discriminance (H5); and concurrent validity (H6).

5.5 Method

This involved a trial of the instrument with three cohorts of students at the conclusion of their first programming course and an analysis of their responses using a confirmatory factor analysis technique (see [HBBA10]).

5.5.1 Data Collection

The sampling frame for each cohort was set to all students who had submitted at least one assignment or code review to ensure participants had indeed attended the course. The minimally adequete sample size requirements was calculated using Cochran's formula for continuous data with finite population correction and adjusted for anticipated non-response [BKH01]. Additionally, this was compared to Westland's Sample Size Calculator tool to determine whether an adjustement would be needed on other statistical grounds (e.g., insufficent number of cases for factor analysis) [Wes10].

A random sampling procedure was used to select participants. Data was collected in three rounds: a paper-based survey was distributed to all students in the lab environment (unselected cases are not considered in analysis); a digital version was then advertised on the virtual learning environment and email alerts were distributed to those whom had not responded to the paper-version; after ten days, an additional series of follow-up emails were distributed to the non-respondents. All participants were offered an opt-out for further communication at each stage.

From 126, 115 and 98 invitations for each respective cohort, 91, 84, and 64 responded. This represents an overall response rate of 70%, noting that 34 cases in 2011-12, 30 cases in 2012-13, and 21 cases in 2013-14 were classified as late respondents. This is because their response was elicited after considerable follow-up during a third round of data collection.

5.5.2 Participants

Participants were all first-year undergraduate students following the sequential pathway for either 'Computer Science' or 'Business Computing'. The descriptive statistics show that less than 20% of the respondents were female, while the average age was 19.5 years, with approximately 15% respondents being mature students (over the age of 23 at entry).

Admission to the pathway required at least 300 UCAS Points (University & College Admission System Points), with a strong preference for STEM subjects (science, technology, engineering, and mathematics). Prior programming experience was not required. However, students without a relevant STEM qualification, or the required points, could opt to pursue a relevant foundation course.

During the introductory programming course, students would learn object-orientated design and the fundamental constructs of the Java language. This was conducted through a sequence of laboratory-based assignments and a collaborative project. The assignment for the 2011-12 cohort was a website and a lab-based programming examination. The assignments for the 2012-13 and 2013-14 cohort were robot scripting tasks, where students would program robots to complete activities such as maze navigation or communication in Morse Code. These assignments were examined by code review and oral viva.

5.6 Data Analysis

The data was analysed in PASW v20 and AMOS 21. All data was analysed. Items under consideration were modified to reflect feedback received from the 2011-12 cohort. As a result items are analysed on a pair-wise basis. This section follows the factor analysis procedure outlined by Hair *et al.* [HBBA10].

5.6.1 Descriptive Statistics

Descriptive statistics for the three samples are shown in Table 5.1 on the following page. This shows that learners tended to report high DSE, PSC, and INT. Many reported low ANX and, as indicated by low APT, many endorsed a growth view of programming aptitude.

Some analyses require the distribution of the data to follow a normal distribution. This was verified through an examination of skew and kurtosis, with skew indices greater than 3.0 and kurtosis indices greater than 10.0 often indicative of severe non-normality [Kli05]. Table 5.1 shows these indices are within these guidelines.

Table 5.1: Mean, Standard Deviation, Skewness and Kurtosis of the Instrument Items

Item	Item Description	*M*	*SD*	*Sk*	*K*
Debugging Self-Efficacy					
DSE1	I am confident that I can understand Java exceptions (e.g., NullPointerException)	3.65	0.96	-0.27	-0.61
DSE2	I am confident I can solve simple problems with my programs	3.48	1.02	-0.18	-0.55
DSE3	I am confident I can implement a method from a description of a problem or algorithm	3.87	0.98	-0.68	-0.26
DSE4	I am confident I can debug a program that calculates prime numbers	3.68	0.92	-0.35	-0.37
Programming Self-Concept					
PSC1	I am just not good at programming	2.44	1.18	0.44	-0.67
PSC2	I learn programming quickly	3.42	1.11	-0.28	-0.58
PSC3	I have always believed that programming is one of my best subjects	3.41	1.17	-0.28	-0.82
PSC4	In my programming labs, I can solve even the most challenging problems	3.34	1.10	0.07	-1.01
Programming Interest					
INT1	I enjoy reading about programming	3.66	1.10	-0.44	-0.50
INT2	I do programming because I enjoy it	3.93	0.95	-0.75	0.13
INT3	I am interested in the things I learn in programming classes	3.72	0.98	-0.52	-0.39
INT4	I think programming is interesting	3.87	1.03	-0.72	0.67
Programming Anxiety					
ANX1	I often worry that it will be difficult for me to complete debugging exercises	2.77	1.06	-0.27	-0.85
ANX2	I often get tense when I have to debug a program	2.83	1.18	-0.08	-0.95
ANX3	I get nervous when trying to solve programming bugs	2.82	1.16	0.06	-0.96
ANX4	I feel helpless when trying to solve programming bugs	2.76	1.21	0.11	-0.83
Programming Aptitude Mindset					
APT1	I have a fixed level of programming aptitude, and not much can be done to change it	2.08	0.97	0.82	0.35
APT2	I can learn new things about software development, but I cannot change my basic aptitude for programming	2.22	0.95	0.34	-0.59
APT3	To be honest, I do not think I can really change my aptitude for programming	1.90	0.92	0.87	0.23

Note: Pooled Sample (N = 175); M: mean, SD: standard deviation, Sk: skew, K: kurtosis.

5.6.2 Measurement Model

To verify the structure of the items for the proposed measurement model (i.e., checking that it was appropriate to group variables together into meaningful constructs), the proposed five-construct solution was evaluated using maximum-likelihood confirmatory factor analysis. As advised in [HBBA10], several fit indices were used to determine fit. One APT item was eliminated at this stage due to a low regression weight. Modifications were also made based on the modification indices to improve overall fit. These fit indices for the final set of items, shown in Table 5.2, indicate that the hypothesised model was 'not a bad fit' to the data (i.e., accepting the null hypothesis of having no significant difference between the prediction and the data).

This suggests that the expected model was adequately reflected by the structure of the data. However, it should be noted that alternative models with superior fit could still exist. An exhaustive review of alternative candidate models is beyond the scope of this chapter.

Table 5.2: Fit Indices and Criteria for the Measurement Model

Fit Index	Measurement Model	Adequate Fit Criteria [HBBA10]
$\chi^2(df = 153)$	267.312	N/A
χ^2/df	1.747	< 3.00
p	0.000	> 0.05
NNFI	0.950	> 0.90
CFI	0.960	> 0.90
SRMR	0.044	< 0.08
RMSEA	0.056	< 0.08

Note: df: degrees of freedom, NNFI: non-normed fit index, CFI: comparative fit index, SRMR: standardised root mean square residual, RMSEA: root mean square error of approximation.

5.6.3 Reliability

Reliability is assessed through examining the Composite Reliability (CR) of each construct. Values close to 1.0 indicate reliablilty, with 0.7 considered minimal [HBBA10]. Table 5.3 shows that the values are consistently above 0.7. Thus, the measurement instrument was reliable with this sample.

5.6.4 Construct Validity

In order to establish construct validity, each construct should demonstrate convergent and discriminant validity. Adequate convergent validity is demonstrated by an Average Variance Extracted (AVE) greater than 0.5 [HBBA10]. Table 5.3 shows all values were above this threshold. Adequate discriminant validity is demonstrated by the $\sqrt{}$AVE being greater than any correlation with another construct [FL81]. Table 5.3 shows that the $\sqrt{}$AVE of each construct was greater than its most significant correlation with another construct. Subsequently, these results imply construct validity.

Table 5.3: Construct Validity of the Latent Constructs in the Measurement Model

	Loadings	Reliability	Variance Explained			Correlations				
Items	*FL*	*CR*	*AVE*	*MSV*	*ASV*	DSE	PSC	INT	ANX	APT
Debugging Self-Efficacy		0.868	0.624	0.530	0.418	(0.790)				
DSE1	0.776									
DSE2	0.808									
DSE3	0.696									
DSE4	0.870									
Programming Self-Concept		0.703	0.655	0.494	0.413	0.703	(0.809)			
PSC1	-0.710									
PSC2	0.800									
PSC3	0.882									
PSC4	0.835									
Programming Interest		0.842	0.579	0.530	0.363	0.728	0.686	(0.761)		
INT1	0.781									
INT2	0.847									
INT3	0.846									
INT4	0.522									
Programming Anxiety		0.888	0.664	0.475	0.364	-0.681	-0.689	-0.507	(0.815)	
ANX1	0.817									
ANX2	0.762									
ANX3	0.834									
ANX4	0.844									
Programming Aptitude Mindset		0.865	0.682	0.262	0.214	-0.431	-0.462	-0.439	-0.512	(0.826)
APT1	0.782									
APT2	0.858									
APT3	0.836									

Note: Values on the diagonal represent $\sqrt{AVE}$; FL: factor loading, CR: composite reliability, AVE: average variance explained, MSV: maximum shared variance, ASV: average shared variance, DSE: debugging self-efficacy, PSC: programming self-concept, INT: programming interest, ANX: programming anxiety, APT: programming aptitude mindset.

Table 5.4: Regression Results for Relations in the Proposed Structural Model

Relationship	Estimate	Standard Error	Critical Ratio	p
APT → ANX	0.310	0.084	3.685	< 0.001
INT → ANX	-0.020	0.149	-0.135	0.893
PSC → ANX	-0.534	0.085	-6.301	< 0.001
ANX → PRACT	-2.335	0.389	-6.004	< 0.001

Note: APT: programming aptitude mindset, PSC: programming self-concept, ANX: programming anxiety, PRACT: frequency of programming practice.

5.6.5 Concurrent Validity

Adequate concurrent validity is established through a cursory examination of the correlation matrix and an examination of hypothesised relationships in a structural model. Table 5.3 does not show any anomalies within the correlation matrix. As such, the proposed structural model was assessed along with a self-report measure of programming practice. The results, shown in Table 5.4, reveal that most of the expected regression relationships were statistically significant. However, the regression between INT and ANX was not statistically significant. This suggests that either there is no relationship or the size of effect is small. Nevertheless, with the exception of INT, the conceptual model appears to be valid.

5.7 Discussion

Adequate measurement in computing education research is important. This is because researchers need to know whether the measures being selected and used by other researchers are valid. Straub *et al.* [SBG04] highlight several key concerns that researchers may have: Does the instrument truly represent the essence or content of the target construct? Is the instrument unidimensional and therefore only representing the target construct? Has the target construct been confused with another similar construct? Are the estimates of the true values of latent constructs appropriate? Rigorous approaches to measurement address such questions.

Unfortunately, there has not been a strong history of reporting psychometric information in the field [RJSL08a] and few measurement instruments are readily available to researchers in the computing education community. Particularly, measurement instruments that capture constructs concerned with the affective-domain of learning computer programming. This may be because developing adequate measurement instruments can be fraught with difficulties [SBG04]. However, there is a strong case for pursuing such work [SBG04, TD13] and there is a range of literature which can be drawn from for support (e.g. [DeV12, HBBA10, SBG04]).

This chapter describes the assembly of one such measurement instrument and demonstrated that it has adequate psychometric properties in terms of reliability, construct validity, and concurrent validity. This instrument focuses on student self-beliefs in the introductory programming context and measures five different constructs: programming aptitude mindset, programming self-concept, debugging self-efficacy, programming anxiety, and programming interest.

It is interesting to note that interest in software development did not predict programming anxiety. In hindsight, other value appraisals such as ‘importance of programming for future prospects’ may have been more appropriate for anxiety.

Nevertheless, programming self-concept and mindset towards programming aptitude were shown to be related to programming anxiety and, subsequently, programming practice behaviour. These relationships have not been firmly established as causal relationships nor are the directions of the relationship clear. This suggests that the measurement instrument will be useful for future work investigating hypotheses raised by the theory in, for example, longitudinal survey studies.

The measurement instrument may also be useful in other similar areas of work. There is a vast range of techniques which educators could attempt to apply in order to enrich their students' beliefs, practice, and performance (see [Mac14]). Using a validated instrument, such as the one proposed here, will improve the rigor of such explorations. The ongoing development of this measurement instrument will support future experiments, increasing confidence that such design experiments present useful and meaningful conclusions. Other uses of the measurement instrument may include educators using the measurement instrument to identify potential problems in their introductory programming classes or researchers evaluating student outcomes across different course designs and cohorts.

5.8 Limitations

It should be noted that this work only represents a first step and future development is needed to overcome a number of limitations. Most importantly, the instrument has only been administered to students at a single institution. Therefore, it may not generalise to populations from other higher education institutions; particularly, those with a different culture. Therefore, there is a need to further validate the tool beyond the institution. Of particular note, the cross-cultural validity of the measurement instrument also needs to be considered in addition to the appropriateness of adapting the framework for different educational contexts. In its present form, it is not clear whether the instrument would be suited for a range of programming topics or age groups.

A small number of items have been included in this scale to facilitate the collection of data from a large group with a short questionnaire. As such, it should not be used to make fine-grain judgements about any individual student. However, estimation of the true values of the latent constructs for individual students would likely improve with additional items.

5.9 Summary

Valid measurement is important, however only a small number of validated measurement instruments are available to computing education researchers. This limits research being conducted into educational theory, teaching practice and the the use of instructional technologies which aim to enrich beliefs and learning behaviour. The study presented in this chapter contributes to this gap in the literature through the assembly and validation of a measurement instrument that could be used for such research. Specifically, for the investigation of student self-beliefs within the introductory programming context.

Three administrations of the instrument at the author's institution demonstrated that the proposed measurement model had a good fit to the data. Furthermore, there was adequate support for reliability, construct validity, and

Table 5.5: Summary of Findings and Adjusted P-Values

RQ	**H$_n$**	**Hypothesis**	$\tilde{p}$	**Conclusion**
5	H$_1$	$FaceValidity$	—	*Supported*
	H$_2$	$ModelFit$	.136	*Supported*
	H$_3$	*Reliability*	—	*Supported*
	H$_4$	$ConvergentValidity$	—	*Supported*
	H$_5$	$DiscriminantValidity$	—	*Supported*
	H$_6$	$ConcurrentValidity$	—	*Supported*

Note: Not all hypotheses are associated with a null-hypothesis significance test.

concurrent validity. This is shown in Table 5.5. However, there are a number of limitations. Critically, the results may not generalise to different age-groups, cultures or educational contexts.

The following chapter will attempt to validate the conceptual framework in addition to exploring appropriate descriptive statistics across two versions of the introductory programming course delivered at Brunel University London. This will support further research into teaching practice and instructional technology used in introductory programming.

Declaration

Some of the work presented in this chapter can also be found in the following publication:

Scott, M. J. and Ghinea, G., Measuring Enrichment: The Assembly and Validation of an Instrument to Assess Student Self-beliefs in CS1, *Proceedings of the 10th Annual ACM Conference on International Computing Education Research (Glasgow, Scotland)*, August 2014, pp. 123–130

6

On Self-Beliefs, Emotions, Practice, and Robots

This chapter extends the work in the previous chapters used to develop a domain-specific approach to establish the consistency of the model. A review of two different introductory programming courses at the authors' institution is presented: a web-focused course run in 2011-12 and a robot-centred course run in 2012-13. Although hours of practice and learning outcomes improve in the robot-centred course, there are no significant differences in the learning experience in terms of programming self-beliefs or programming anxiety.

6.1 A Comparison Between Two Courses

Previous chapters have highlighted the importance of domain-specific forms of self-concept and mindset for student learning behaviour. Additionally, as limited work has been conducted in specialised fields such as computer programming, a measurement instrument has been presented to further this line of research.

However, it is important not to privilege an *internal view* of the causes of practice behaviour and to recognise the *external view*. That is, the role of situation and context. To do otherwise would be a fundamental attribution error. While a review of external factors that influence practice behaviour is beyond the scope of the current study, there is sufficient data to examine the consistency of the proposed model across different educational practices and to examine their respective influences on the key variables in the model (i.e., self-concept, mindset, anxiety, and practice).

Key differences in the design of learning environments are often centred upon learning activities, assessment, and feedback. Many educators provide students with worksheets. They then provide encouragement and rich feedback to motivate students *as* they encounter challenges in practical laboratory sessions [Jen01]. However, such efforts are limited and may not encourage practice beyond the laboratory environment. An approach that shows promise beyond the laboratory environment, however, is the use of personal robots [Kay10], whereby each student is provided with their own programmable robot. In theory, the use of robots makes learning activities more engaging, motivating students to spend more time experimenting with their programs [MK10]. They also provide a more intuitive source of feedback as they reinforce mental models in a visual way [ALM$^+$10], helping students to fix problems and overcome frustrations in a relatively short time.

However, while the potential impact of robots is promising, it is not clear how their use interacts with students beliefs, emotions and practice behaviour. This is key to preparing an appropriate context for learning that will maximize their potential impact. That is, robots are simply tools that support, complement and enhance learning environments. As an example, robots frequently grab attention [MK10, LNH09, ALP10]. Thus, robot can draw upon students' curiosity to engage them with an assignment. However, there is little evidence to suggest that the mere presence of a robot makes an assignment more relevant, better able to inspire confidence, or more satisfying [McG12].

This chapter evaluates a course centred around a Robot Olympics against a more traditional web-programming course. In this introductory programming course, students learn to program their own personal robots during worksheet-based laboratory sessions. To engage students in practice that leads to improvements in the quality of their code, their experimentation with their personal robot is supported through regular code reviews. This prepares students for an end-of-course event—the Robot Olympics itself—where they program their own robots to complete a specific task in a dedicated space as an assessed demonstration.

6.2 Related Work

The use of robots in educational contexts has grown in popularity since the early millennium [FM02], and robot-centred courses are often positively received by students [MK10]. Furthermore, a systematic review has shown that robots, in general, can be effective when teaching computer programming [MKB12]. However, there are questions about the effectiveness of robot-centred courses, highlighting a need to explore achievement data [Ali13].

Where such evidence is available, robot-centred programming courses have not consistently demonstrated success. A study conducted at the US Air Force Academy found that scores in a robotics section of a programming course were lower than in a non-robotics section [FM02]. Limited access to the robots has been suggested as the potential reason for this result, as students had insufficient time to reflect and engage in further experimentation [FM02]. Consequently, the availability of robots could moderate the effectiveness of a robot-centred course.

Now that low-cost robots such as the Finch [LN10] are available, students can learn using their own personal robots. In order to motivate students to engage in programming practice with their robots, various strategies can be used [Rob00]. A popular choice is hosting a Robot Olympics [Mur00, CV13, LH05, VA04]. These 'games' often take the form of events where students demonstrate solutions to a set of well-defined tasks. For example, students in the Trinity College Fire-Fighting Home Robot Contest explore programming through the development of a robot that can navigate a mock home to extinguish a candle [VA04]. This approach would seem to represent a novel and engaging learning environment. However, the method has not been formally evaluated as a model for an introductory programming course, so the potential impact is unclear.

It is important to clarify this impact to help those deciding to introduce robots into a course. For example, Kumar questions "is it worth using robots for traditional projects in [an] AI course?" and concludes "no, if we consider the time and effort that robot projects demand" [Kum04]. Such demands range from storage, locating spare robots when students forget them, diagnosing abnormal robot behaviours, repairing mechanical failures, and the increased duration of assessments; which, may not be justified if any positive impacts are marginal.

6.3 Intended Outcomes

The impact of a robot-centred programming course is evaluated through comparison with a reference group. The reference group was drawn from a previous year on the same program within the authors' department. Both groups had the same entry requirements and followed a similar structure. As typical for a UK institution, each involved a year-long course structured across two terms of twelve weeks. However, the earlier cohort focused on web programming as the robots had not been introduced at that point into the department. As such, the following research questions were posed in order to determine differences between the two cohorts:

RQ1. Will the students enrolled on the course leading to the Robot Olympics report different self-concept, or anxiety compared to those enrolled in the web programming course?

RQ2. Will the students enrolled on the course leading to the Robot Olympics report spending more hours per week practising their programming skills compared to those enrolled in the web programming course?

RQ3. Will the students enrolled on the course leading to the Robot Olympics produce code of an overall higher quality compared to those enrolled in the web programming course?

RQ4. Will greater levels of practice and practice within the context of preparing for a Robot Olympics predict aspects of code quality differently?

As it is possible that robots may not be equally effective for all students [Froyd, 2013, personal communication], gender differences are also explored. Broadening participation is an important goal for computing education research because of under-representation in the workforce stemming from *the pipeline shrinkage problem* [GC02, MF03] with some work showing that female high school students are less likely than male high school students to want to pursue computing [Pap08]. It is, then, important to ensure that robots do not further undermine participation in computing programmes. The first research question therefore examines two hypotheses: male and female students enrolled on the robot-centred course will report more hours of practice per week ($H_{1\text{-}2}$). The second research question examines three hypotheses: that practice will have a direct effect on overall code quality (H_3); that enrolment on the robot-centred course will have a direct effect on overall code quality (H_4); and that gender will have a direct effect on overall code quality (H_5). The third research question addresses three core hypotheses: that gender, course enrolment and more practice will have different impacts on three different aspects of code quality, namely functional coherence, readability, and sophistication. As this represents three variables of interest and three aspects of code quality, this results in an additional nine hypotheses ($H_{6\text{-}14}$).

6.4 Course Design

Both courses were practical introductions to computer programming that expect students to:

LO1. Demonstrate an understanding of the basic concepts of programming

LO2. Analyse a problem and produce a computer program as a solution to that problem

LO3. Use a simple development environment to produce viable program code

As both cohorts were supervised by a similar team of core teaching staff, and the robot-centred course built upon existing teaching practice within the department, both courses followed similar basic structures. This consisted of a series of lectures introducing concepts to students and laboratory sessions which then reinforce those concepts through practical programming tasks (similar to [Gei94]). Laboratory sessions were organized weekly to ensure regular practice. The web programming cohort had mid-term examinations, while those preparing for the Robot Olympics had their code formally reviewed by teaching assistants. This meant that all students received ongoing soft scaffolding [SK07b] and feedback at regular intervals [GS04] as this strategy is believed to be more effective for encouraging practice [BED+03].

Both cohorts completed their respective programming tasks as part of a group project. Thus, students were divided into groups of five or six. Meetings with group tutors facilitated individual support and helped to prompt students to reflect on their progress through small group learning activities [SSD99]. This strategy also helps students form learning communities, encouraging mutual support during challenging tasks and transforming experimentation with robots into social learning opportunities [Ver13, FO05]. It also reflects and presents (in a small way) the composition and communication issues found in the IT industry [She11b].

While there are many similarities between the two courses, there are also a number of key differences. Table 6.1 presents these differences, showing that the courses are comparable but also highlighting several contrasts. The earlier cohort focused exclusively on web programming and the later cohort on the use of the robots. As the learning objectives were the same, there were many similarities between the tasks each group member had to undertake: firstly, both had to be written in Java; secondly, both had to demonstrate the same range of programming constructs; thirdly, both examined how students separated the user interface (i.e., presentation layer) from the key functionality (i.e., domain logic layer); finally, both validate user input and both demonstrate file processing. The assessed problems were also designed to be of similar size and complexity. However, the robot coding problems were different in nature, as they were designed to capture students' interest and make use of the robots' capabilities[1].

The web form processing tasks, such as adding content to a file and displaying it on a web page, were replaced with 'events' within the Robot Olympics. Examples include: Morse code communication, where the robot would use the light on its beak to communicate Morse code translations; an obstacle course,

[1]Further details on the Robot Olympics and its assessment method are available at: http://dx.doi.org/10.13140/2.1.3680.9281

Table 6.1: Key Differences between the Web Programming Course and the Robot-Centred Course

Course Element	Web Programming	Robot Olympics
Personal Robot	✗	✓
JavaServer Page Project	✓	✗
Robot Olympics Project	✗	✓
Assessed Worksheets	✗	✓
Exams	✓	✗
Oral Viva	✗	✓
Python Classes	✓	✗
Java Classes	✓	✓

where the robot had to navigate across an arena; a robot controller, where the movement of the robot is controlled directly through a user interface; and tunnel navigation, where the robot measured the lengths of tunnels with its light sensor.

Another key difference is that the web programming cohort followed a combined Python and Java curriculum which focused on JavaServer Pages (JSP) (see [JLTS11, KLMss] for details). As such, new learning content replaced some of the previous material (i.e., Python and JSP classes) in order to help students with the new mode of assessment (i.e., using the robots).

6.5 Method

6.5.1 Data Collection

For measuring self-beliefs and practice, the same data that was collected in the study reported in Chapter 5 was used. Extending this data a measure of student performance was incorporated into the dataset. Coursework submissions for the introductory programming course were provided by the respective module leaders (which were downloaded from the archive in the e-learning environment and had not been modified in any way). The sample was matched to that used in Chapter 5, such that only the coursework submissions of students who participated in the prior data collection activities were used.

6.5.2 Research Instruments

6.5.2.1 Student Self-Beliefs

The questionnaire items and model prepared in Chapter 5 were used to impute scores for programming self-concept, programming aptitude mindset, and programming anxiety.

6.5.2.2 Self-Reported Weekly Programming Practice

Hours of programming activity per week was assessed using a self-report measure. The item "in a typical week during term-time, how frequently did you write code and/or work on programming related activities?" was presented as a 7-point Guttman-style item. Each response option was labelled "at least $\{x\}$ hours" where x increased in multiples of five.

6.5.2.3 Code Quality

Quality of coursework submissions was scored according to a marking scheme by a single rater. Although tasks were different, both shared a common set of learning objectives and level of sophistication. Three aspects of code quality were assessed: *functional coherence*, which measured whether solutions successfully implemented the set requirements; *readability*, which measured whether the solution was commented and structured appropriately for future maintenance; and *sophistication*, which measured whether an appropriate range of programming constructs had been used. Each aspect was scored according to five descriptors, with the lowest indicating inadequate quality. Moderation of 12 truncated submissions showed that self-consistency ($\alpha = .91$) and faculty agreement with marks ($\alpha = .72$) were adequate [HK07].

6.6 Data Analysis

The data was analysed using PASW 18.0.3 and AMOS 21 for Windows. Cases with missing data were excluded list-wise. All p-values from null-hypothesis significance tests are two-tailed with statistical significance being determined at the conventional level (i.e., $\alpha = .05$).

6.6.1 Differences in Attitude

Multiple factorial (2 x 2 x 5) between-subjects MANOVA evaluated the impact of the robot-centred course on each student self-beliefs for each gender and level of practice reported. As anticipated, the multivariate effect of practice was significant ($F = 2.509, p = .012, \eta_p^2 = .068$). There were no other significant multivariate effects, however it should be noted that a complex interaction between gender, level of practice, and course approached significance ($F = 2.328, p = .056, \eta_p^2 = .068$). Given that there is low power for detecting gender effects, this warrants further exploration in the future.

The contribution of each parameter to the model is shown in Table 6.2 below. As no gender and course differences were found in the multivariate test, no additional hypotheses tests were conducted and only the observed differences are shown. The results reinforce the relationship between practice and self-beliefs, but highlight concerns that anxiety and self-concept remained consistent across the course designs.

6.6.2 Greater Practice with the Robot Olympics

As self-reported programming practice did not follow a normal distribution, a Mann-Whitney U Test was conducted to examine the difference between the two cohorts. This indicated that programming practice was greater in the robot-centred course compared to the web programming course for both male students ($U = 1404, p = .016, r = 0.21$) and female students ($U = 73, p = .024, r = 0.40$). This is shown in Figure 6.1, where the proportion of students studying for less than ten hours per week decreases and those studying for more than ten hours per week increases. This represents an overall increase of 37.4% based on the mean difference. However, it should be noted that the median did not change and 54.8% did not fulfil the expectation of 10-15 hours per week.

6.6.3 Higher Overall Quality with Practice and the Robot Olympics

A factorial (2 x 2 x 5) between-subjects MANOVA evaluated the impact of the robot-centred course on the quality of student submissions for each gender and level of practice reported. Assumptions of normality were supported. However, the Box's and Brown-Forsythe tests only supported equality of variance and equality of the covariance matrices once readability was collapsed into four categories (rather than five). As the cell sizes were not equal, Pillai's Trace was used to assess significance. Significant multivariate effects for practice ($Trace = .190, F = 2.202, p = .011, \eta_p^2 = .063$) and for course ($Trace = .134, F = 6.603, p < .001, \eta_p^2 = .134$) were found. This is illustrated in Figure 6.2, which shows that overall quality is higher in the robot course and increases with practice. However, there were no significant effects for gender ($Trace = .019, F = 0.829, p = .480, \eta_p^2 = .019$).

6.6.4 Varying Effects of Practice and the Robot Olympics on Aspects of Code Quality

To examine the effects on each aspect of code quality, univariate ANOVAs were conducted. These are shown in Table 6.3. As no gender differences were found in the multivariate test, no hypotheses tests were conducted and only the observed differences are shown. The results suggests that preparation for the Robot Olympics helped students produce higher quality code in terms of functional coherence and sophistication. Furthermore, programming practice predicted functional coherence. Interestingly, however, programming practice itself did not predict higher code readability or sophistication. Figure 6.3 shows a pair of scatter matrices illustrating the correlations between practice and each aspect of code quality alongside the potential interaction these correlations have with course design. It is important to note that the first column shows that those with lower levels of practice tended to exhibit increased quality when enrolled on the robot course, suggesting that the robots may mitigate some of the problems associated with low levels of practice. For example, qualitative differences in the experience may have lead to deeper (higher quality) learning.

Table 6.2: ANOVA Results For Self-Concept and Anxiety

Predictors	F	p	η_p^2	d
Programming Self-Concept				
Level of Practice	4.362	.002	.112	
Course	—	—	.001	0.00
Gender	—	—	.008	-0.32
Programming Anxiety				
Level of Practice	3.134	.017	.083	
Course	—	—	.002	0.03
Gender	—	—	.011	0.28

Note: The estimate of Cohen's d was calculated using the estimated marginal means from the MANOVA and the pooled standard deviation of each variable.

Table 6.3: ANOVA Results For Each Aspect of Code Quality

Predictors	F	p	η_p^2	d
Functional Coherence				
Level of Practice	5.321	.001	.141	
Course	19.268	.000	.129	1.15
Gender	—	—	.003	0.27
Readability				
Level of Practice	0.919	.455	.028	
Course	1.504	.222	.011	0.36
Gender	—	—	.016	0.39
Sophistication				
Level of Practice	0.798	.529	.024	
Course	5.426	.021	.040	0.74
Gender	—	—	.011	0.37

Note: The estimate of Cohen's d was calculated using the estimated marginal means from the MANOVA and the pooled standard deviation of each variable.

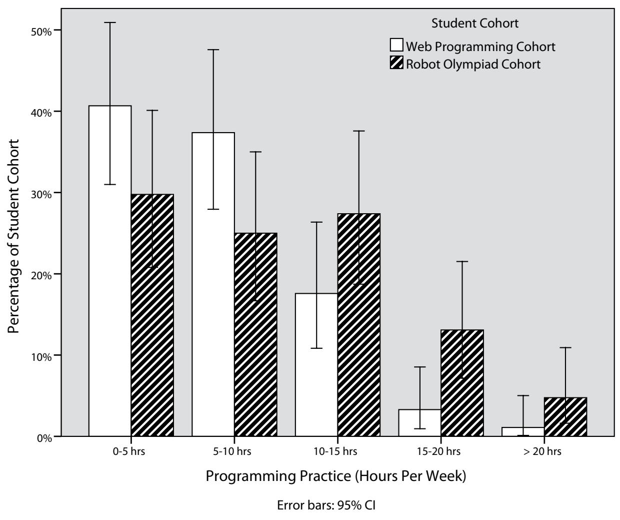


Figure 6.1: A clustered bar chart comparing self-reported hours of programming practice between students enrolled in the Web Programming and Robot-Centred courses.

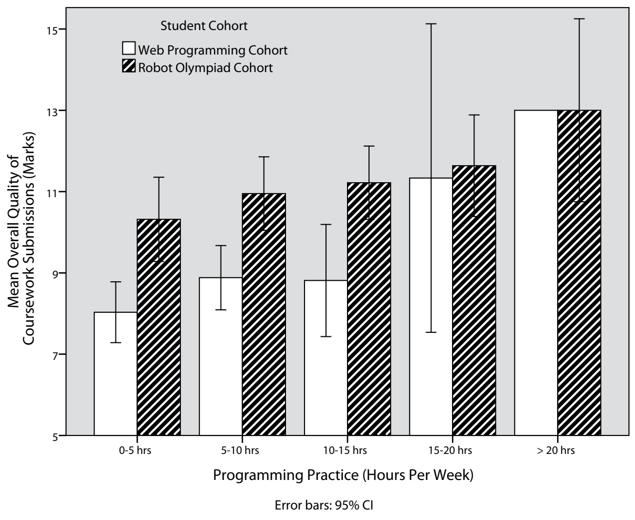


Figure 6.2: A clustered bar chart comparing the overall quality of final coursework submissions between students enrolled in the Web Programming and Robot-Centred courses at each level of self-reported practice.

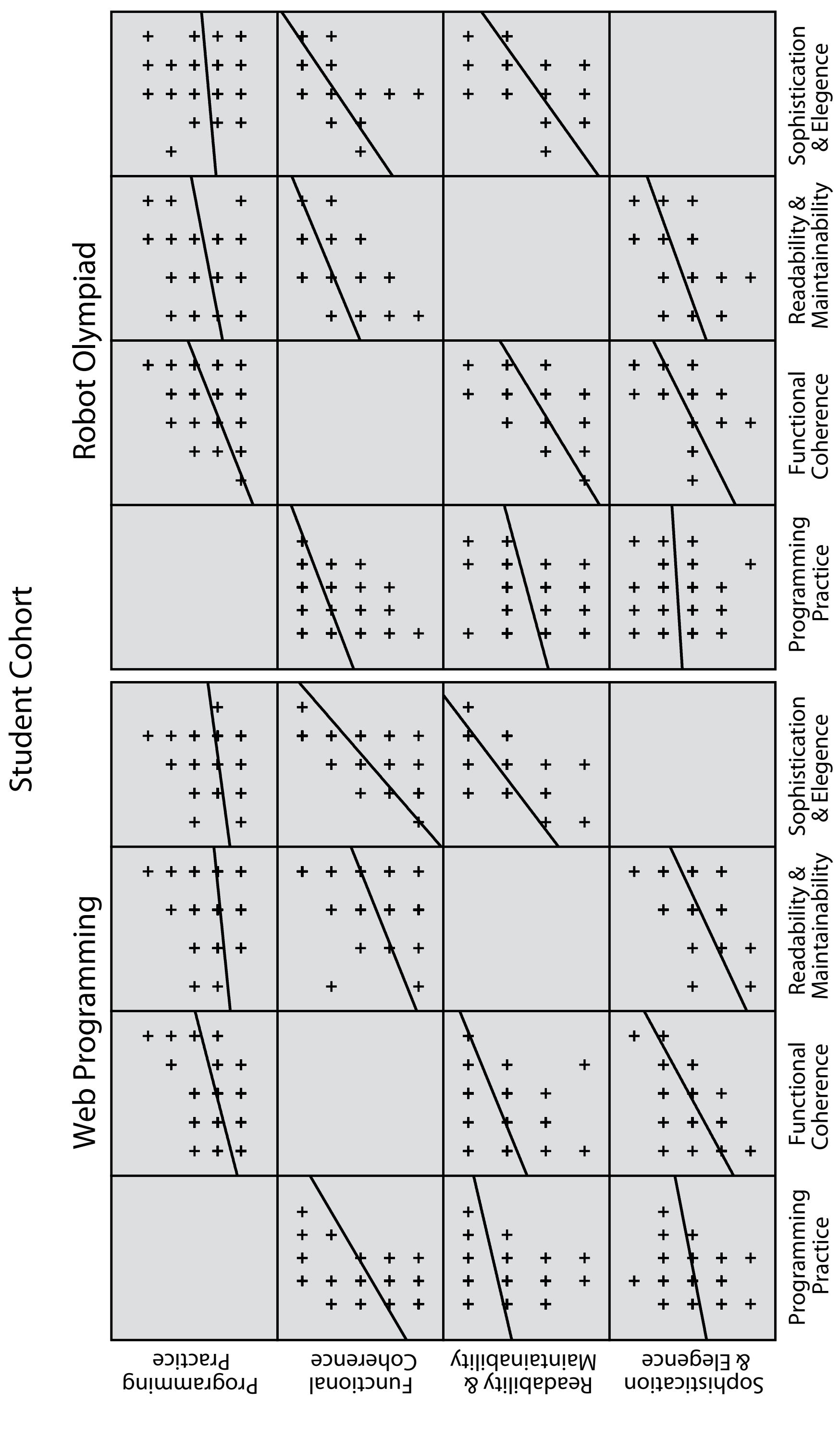


Figure 6.3: A pair of scatter plot matrices comparing the relations between programming practice, aspects of code quality, and student cohort.

6.7 Discussion

The findings reinforce the notion that robot-centred learning environments can encourage programming practice. The frequency of students reporting at least ten hours of practice per week increased from 22% to 45%. Nevertheless, engagement remains an issue with more than 50% of students not pursuing the levels of practice that had been set as an expectation. Further investigation into student practice could result in improvements to the Robot Olympics by, for example, considering other potential influences (see Chapter 2).

The results reveal some evidence which suggests that practice within the context of the robot-centred course can be more effective than alternatives. This is important to consider, because *how* students practice is just as, if not more, important than how long they practice [EKTR93]. Examining overall quality ratings across different levels of practice revealed that those students involved in the Robot Olympics consistently outperformed those on the web programming course at lower levels of practice. However, further research is needed to explain why this is, because the available data cannot isolate the contribution of any individual change to the course, such as the use of personal robots.

Enrolment on the robot-centred course predicted functional coherence, even when accounting for practice as a covariate. This suggests that students fulfilled the requirements more successfully. Hence, introducing personal robots within an appropriate context can lead to improvements in student outcomes in a way that just additional time-on-task does not. An insight is that the physical feedback can be easily understood by students, whereas a small difference in a web page may not be. Thus, the way in which the nature of the coding interface and its feedback supports students' development of mental models warrants investigation.

It was anticipated that the code reviews would help students improve the readability of their code. However, enrolment on the robot-centred course did not predict readability. There are several explanations for this, with one such hypothesis being that students prioritized the functionality of the code as watching robots complete tasks is more compelling. However, the format of the reviews could also be a factor.

Enrolment on the robot-centred course predicted sophistication where practice did not. As such, the challenges presented during code reviews and the robots themselves could have pushed students to improve. It is interesting to note that the increase was supported by a higher proportion of those with low levels of practice receiving higher scores. Perhaps differences in style of cognition, creative thinking, and reflection promoted this increase. Further work is required to isolate and explore these hypotheses.

There were no statistically significant differences in terms of gender. However, there were some potentially meaningful differences in terms of effect size. Most notably, female students showed greater changes in level of practice based on course enrolment than did male students. Additionally, there could be small but meaningful differences in terms of achievement, but this cannot be verified due to the low statistical power associated with *post-hoc* analyses.

6.8 Limitations

The study was observational in nature, so there may be factors unaccounted for. For example, the two cohorts could have differed from the outset in unobserved respects, there may have been differences in teaching quality, and so on. Furthermore, as several changes were made, it is the entire course that is assessed, rather than specific differences. The analysis focuses on quantitative data, excluding potentially useful qualitative data on the use of robots. As such, it is unclear whether their characteristics encouraged different approaches to learning and writing code. Caution should be exercised when interpreting self-report questionnaire data [DGV02b]. Additionally, only those who submitted code for review were included, thus perhaps excluding those students who failed to engage with the robots. Finally, the *post-hoc* hypotheses associated with gender had a high probability of type-II error.

6.9 Conclusions

Practice and reflection both play important roles in the development of programming expertise. As such, it is important to design courses that encourage these activities. This chapter explores a robot-centred approach to promote these activities that was trialled in an actual course at the authors' institution. This reveals some evidence that the use of personal robots in an appropriate context can inspire students to engage in frequent practice. Enrolment on the new course also predicted two aspects of code quality: functional coherence and sophistication. This demonstrates that the robot-centred course improved student outcomes in a way that just additional time on task does not, suggesting a qualitatively different learning experience. For example, did the physical feedback from the personal robot aid in the construction of mental models? As such, further work is needed to evaluate how students' reflective activities, meta-cognition, creativity, attitudes, motivation, and approach to learning changed as a result of participation in the new robot-centred course.

Declaration

Some of the work presented in this chapter can also be found in the following publication:

Scott, M. J., Counsell, S., Lauria, S., Swift, S., Tucker, A., Shepperd, M., and Ghinea, G., Enhancing Practice and Achievement in Introductory Programming with a Robot Olympics, *IEEE Transactions on Education* (Accepted, In Press), pp. 1–6, DOI: 10.1109/TE.2013.2288700

7

Games-based Fantasy Role-Play in the Programming Lab

To some extent, the previous chapters show that beliefs predict emotions and emotions predict behaviour. Hence, it is important to help students overcome non-constructive beliefs so they can engage in deliberate practice. This chapter proposes that games offer opportunities to help educators to achieve this aim. The concepts of procedural rhetoric and narrative reinforcement are introduced as persuasive mechanisms that could enrich students' beliefs. An analysis of 52 games that teach programming is then presented, illustrating that few of them target student beliefs or take advantage of these mechanisms. This invites further research to address the potential utility of these mechanisms for influencing student self-beliefs.

7.1 Introduction

In line with previous research on similar topics [VDC04, JVVDP04], the previous chapters highlight that programming self-concept and programming aptitude mindset can predict anxiety in the programming lab. They also show that anxiety can predict levels of programming practice and, additionally, that practice can predict (some aspects of) performance. Assuming that these relationships are causal in nature, following the Control-Value Theory of Achievement Emotions [Pek06], it would seem pertinent to enrich these self-beliefs in order to help students manage their anxiety and subsequently help them to engage in deliberate practice regularly. However, analysis of outcomes across three previous cohorts at Brunel University has shown that the transition from an exam-based web programming course to a practice-based robotics course had no statistically significant effect on anxiety. Thus, this emotive dimension of learning computer programming has not been adequately addressed by existing teaching practices.

There are a number of practices which could be applied to address the affective domain. For example, growth messages can be embedded into assessment rubrics and feedback [CCD+10], soft-scaffolding can help students to develop a stronger self-concept [OMCD06], and students can be encouraged to engage in work that they can take pride in [GCF+10]. There are, however, some limitations associated with these approaches. In particular, educators in the UK often now need to manage the scalability of their teaching due to increasing administrative demands [Tig10] and staff-student ratios [Cou12]. Typically this is achieved through the support of automated marking systems, which may not be suited to providing the necessary types of tasks and feedback, and through the support of teaching assistants, whom may not receive the necessary preparation needed to implement these interventions. An alternative approach would, therefore, be welcome in large classes. One such approach is to embed the intervention directly into new educational tools that educators use with large group teaching and one type of tool that shows promise, in this respect, is digital games.

7.2 Digital Games in Educational Settings

Digital game-based learning refers to the use of instructional technology to produce a synergy between fun and learning, often induced through tightly coupled cycles of action and feedback situated within a "magic cirlce"[1]. Research on the potential of games-based learning has continued throughout the past 35 years, growing from early work on the psychology of video games [LL83] and the serious applications of games [Abt87]. Malone and Lepper's work during the 1980s [Mal80, ML87] and their students' work in the 1990s [PL92, CL96] highlighted the intrinsically motivating aspects of fantasy games and since then an ever-increasing body of literature has evolved around discourse on the educational and psychological potential of digital games (e.g., [RMWW92, D$^+$94, Rie96, Kaf01, RM01, Squ03, HAB05, Hay05, DF06, Sha06, VVCB$^+$06, Don07, EN07, FT06, Ke09, Squ11, HCSB11, DSN$^+$11, HH$^+$11, TFDW11, GS12, YSC$^+$12]).

[1]Castronova defines this as the "shield of sorts, protecting the fantasy world from the outside world" [Cas08, p. 147]. However, refer to Huizinga [Hui86] for a more comprehensive introduction to the term and Suits [Sui14] for further philosophical engagement with this notion and an alternative term "lusory attitude".

In more recent years, digital game-based learning has been increasingly evangelised as a method of teaching and self-improvement (e.g., by Prensky [Pre05], Gee [Gee03, Gee14], Ritterfeld [RCV09], Schell [Sch10], Zichermann [ZL10], McGonigcal [McG11a, McG11b], Sheldon [She11a], etc.). There has been criticism of the lack of evaluation of learning games [RMWW92, CSH07, TFDW11] and the different approaches taken to gather evidence [CBM+12]. Additionally, not all of the evidence supports their use [AMM+12]. Nevertheless, there are several meta-analytical reviews suggesting that using digital games for instruction has some value [Pet09, Sit11, CBM+12, GEM13, WVNVOVDS13]. In particular, some games have been shown to be effective for general computing instruction from a learning perspective [Pap09, HCSB11].

The games which are successful at instruction tend to have clearly defined rules, objectives and expected outcomes. They also tend to provide rich feedback, present an engaging narrative, provide scaffolding through appropriate levels of challenge in order to drive their deliberate practice. They do, then, take advantage of educational principles to improve engagement and retention throughout the learning processes. An area in which games have been shown to be considerably effective is in applied interdisciplinary contexts, whereby students utilise a range of knowledge, critical thought, and problem solving towards goals that they find interesting [SVKT08]. However, the capabilities of games to manipulate self-beliefs through persuasion is not clearly described or evaluated in the educational games literature. Instead, the persuasive mechanisms that exist in games can be derived through examining the literature of media communications and psychology (i.e., experimental research on media effects) alongside game studies (i.e., the use of games as forms of rhetoric).

7.3 Persuasive Mechanisms in Game-based Fantasy Role Plays

The mechanisms that games can use to persuade are varied and multifaceted. As with many other forms of multimedia presentation, rhetorics associated with the oratory and visual traditions can be employed. However, unlike these other media, digital games distinguish themselves in two key ways. Firstly, games are highly interactive by definition. This means that players take an active role in systems and environment that will communicate with them and respond to their actions; thereby, providing multiple opportunities for persuasion through direct feedback to actions. Secondly, games are immersive in nature. They enable players to assume a role that transform their identity; hence, providing a lens which helps them to consider new perspectives on themselves and the world they inhabit. Hence, games present two different sets of affordances, falling under the broad classification of *procedural rhetoric*, where persuasion is embedded within process and system, and *narrative reinforcement*, where persuasion is embedded within representation and narrative. An overview of the methods is shown in Tables 7.1–7.2.

Table 7.1: Types of Procedural Rhetoric used as Persuasive Mechanisms in Game-based Fantasy Role Plays

Type of Mechanism	**Description**	**Sources**
Embodiment	Use of a virtual environment to transform the perspective and identity of a participant using an avatar	[YB07], [PSAS13]
Vicarious Reinforcement	The manipulation of the characteristics of an avatar in response to participant actions	[FB09]
Gamification	Use of meaningful token economies and game mechanics to quantify and reward participant actions	[ZL10], [She11a]
Process Meaning	The integration of meaning and argument into game mechanics and other processes	[Bog07]

Table 7.2: Types of Narrative Reinforcement used as Persuasive Mechanisms in Game-based Fantasy Role Plays

Type of Mechanism	**Description**	**Sources**
Imago Effect	The use of a narratively rich character as an avatar	[THB08], [Gee14]
Adventure Programming	The integration of novel challenges (with parallels to real-world problems) into interesting environments	[Han00]
Emotion Design Patterns	The use of story, narrative and appropriate mechanics to evoke positive emotions	[Laz04], [Fro07], [Swi08], [DWN13]
Agent-based Feedback	The use of non-player characters to provide encouragement and support	[LK11]

7.3.1 Procedural Rhetoric

Gee [Gee03] considered the role of identity in learning. A three-way relationship was proposed between: real-world identity; virtual identity; and a mediating projective identity. It was argued that the virtual and projective identities formed when immersed in a fantasy role could reinforce the real-world identity relating to the activity of that role. During a separate investigation of self-representation in virtual reality, Yee and Bailenson [YB07] empirically demonstrated a somewhat similar phenomenon relating to identity and self-beliefs: the Proteus Effect. Other studies have also shown that this can influence implicit beliefs [PSAS13]. As such, manipulation of character mindsets and self-concept could correspond with changes in player mindsets and self-concept.

Vicarious reinforcement is a concept that was introduced by Jesse Fox [FB09]. It is a form of reinforcement where the consequences of actions are visualised within the game context. In the first example, players who exercised frequently would see their avatar lose weight while those who exercised seldom saw their avatar gain weight. As such, the visual cue served as a reminder for players to engage in exercise, so as to avoid being represented in a way that they did not feel comfortable with.

The concepts of gamification and gameful design can be used to quantify experiences in terms of progress towards a goal or otherwise providing a score. Part of self-concept development is comparison between ones environment and ones perceived self. However, it may not be the case that novices can make accurate judgements about the quality and value of their work. By providing adequate quantification and demonstrating to players how they are progressing, their development of self-concept can be supported in a more deliberate fashion. That is, they can compare their own ability with externally validated criterion rather than subjective norm-based comparisons against their peers who may have prior experience in programming. There is some empirical support for this technique through its application in a mathematics game [OHB+14], largely enabling players to level up and pushing them to more advanced challenges.

Process meaning is a concept developed by Ian Bogost in his book Persuasive Games: The Expressive Power of Videogames [Bog07]. It is described as "the art of persuasion through rule-based representations and interactions, rather than the spoken word, writing, images, or moving pictures." In this way, the concept can be applied to provide students with growth mindset messages through the way in which the game is presented.

7.3.2 Narrative Reinforcement

The Imago Effect [THB08], also referred to as projective identity [Gee14], is the ability of players to relate to characters and situations that are presented to them. This provides to reflect on their own skills and ability when framed as a character. It is distinguishable from the Proteus Effect as the Imago Effect may not be specific to the player avatar and may not even be subject to manipulation.

Adventure programming refers to a technique widely used in secondary education contexts to improve student's locus of control[2] It refers to the practice

[2]Locus of control is one of the core self-evaluations constructs. These are sometimes collectively referred to as "positive self-concept" constructs [JLDK98, JB01]. This is because these areas of research complement each other.

of exposing students to “real life situations in which they have to employ problem solving or otherwise creative methods to deal with the environment around them and the task at hand” [Han00, p. 34]. In the context of game design, this same concept can be applied to help students apply their skills in a novel fantasy context whereby they overcome a challenge. Such challenges are, typically, analogous to the real-world. An example is *Space Mission: Ice Moon* [DG07], where students are presented with a science fiction context where they are able to apply their science skills to help save a team of astronauts in a disaster scenario. Such role play helps students attach meanings and emotions to their actions, thereby providing motivation to persist and address any non-constructive beliefs they may have about their abilities.

There are a range of emotional design techniques [Laz04, Fro07, Swi08] and design patterns [DWN13] which can be used to effectively as part of an intervention to reduce negative emotions. For example, the narrative and in-game tasks can be designed in such a way so as to avoid triggering students’ anxieties. Additionally, the design of the game can be purposeful so as to facilitate positive emotions that may enhance students’ self-concept. By, for example, providing opportunities for students to feel *fiore* and pride in their efforts as this has a positive influence on self-concept development [GCF$^+$10].

An agent is a computer program that acts for a user on their behalf. In this case, the agents aim to support students in ways a teacher otherwise would. This approach has been shown to be effective; in particular, previous work has revealed that novices find them helpful when learning to use new items of software [LK11] and their self-efficacy can increase as a consequence of encouragement.

Part of effective self-concept interventions relies upon rich feedback and positive reinforcement being provided to students in a timely manner. As such, the use of characters within a narrative setting can help frame feedback and provide a source of encouragement to learners, thereby supporting their development of self-concept.

7.4 Games in the Programming Lab

There have been several approaches to the incorporation of digital games and game-like instructional technologies into the introductory programming laboratory. Li and Watson [LW11] categorise these games into two types: *authoring-based*, taking a constructionist approach which challenges learners to create games and stories (e.g., ALICE [CDP00, KP07]), and *play-based*, taking a guided-instruction approach wherein lessons emerge through systems of play and puzzle solving which demand programming strategies and code to complete (e.g., CodeSpells [EFG13]). Additionally, their review highlights games produced using: a range of technologies (including 2D and 3D); targeting different learners (in terms of age, knowledge, and prior experience); different forms of coding such as typing, using graphical objects, and completing pre-defined forms.

Despite the claims made by authors, such as Gee [Gee03, Gee14] and Yee [YB07], on the transformation potential of games and virtual worlds, there appears to have been a priority to address cognitive and skill-based objectives, rather than affective objectives, in the development and evaluation of serious games. Only 7 of 129 studies surveyed by Connolly *et al* [CBM$^+$12] address affective and motivational factors in educational games. Likewise, in the respective evaluation frameworks proposed by Mayer *et al* [MBH$^+$14] as well as Connolly *et al* [CSH08], affective objectives are omitted. As such, previous reviews do not make it clear whether or not *any* digital games address the enrichment of student self-beliefs, never mind those published specifically for application in computing education. Hence, to remedy this gap in the literature, a review of existing programming games is presented in this chapter.

As there is no exhaustive list of games used in introductory programming classes, and several have been produced by professional game developers rather than academic researchers, a list of candidate games was curated through a literature search in the ACM Digital Library, IEEE Explore, and ScienceDirect. Some additional titles were then curated through the DiGRA, GAMESNETWORK, and IGDA Games Education mailing lists. Other games were drawn from the review by Li & Watson [LW11] as well as a recently published reviews by Vahldick *et al* [VMM14] and Malliarakis *et al* [MSX14]. This resulted in 61 games and game-like products being found. Both primary sources (i.e., the games themselves) and secondary sources (i.e., reviews and articles about the games) were used in the analysis. The following inclusion criteria were also applied to the set:

- Presents a play-based game environment
- Required players to write code or construct a programming artefact (e.g. write code using text or a drag-and-drop block-language)
- Addressed learning outcomes related to basic programming constructs (e.g. control flow and variable assignment)

- Game available to play in English or information available in English

An analysis of the remaining 52 games is shown in Table 7.3 on the following pages. The table shows the name of the game, the type of game classified using *in vivo derived* genre patterns, and whether or not it contained any persuasive mechanisms.

While most of the games possessed some persuasive mechanism (66%), self-belief variables are seldom addressed, with the only game included in this survey explicitly addressing them being Gidget [Lee13], doing so through process meaning and agent-based encouragement.

Nevertheless, approximately 62% of the games analysed contained some persuasive mechanism in the form of narrative reinforcement. Most of these were in the form of Adventure Programming (38%) and often accompanied by the context of an Imago Effect (36%) arising through the use of character-driven narratives which provided context for increasing challenge and mastery-experiences. Additionally, many games were also supported by the use of agent-based feedback (47%) even where no narrative was presented within the game itself. However, it should be noted that the nature of the feedback varied considerably from static encouragement in response to play choices (e.g. in World of Variables) to more expressive and purposeful personifications which aimed to provide rich evaluative feedback (e.g. in Gidget). Surprisingly, few games adopted common emotion-design patterns (14%). The patterns tend to emerge in the most narratively-driven games, which tend to be role-playing games (RPG), however, these were less common. This could be because the most popular types of games were LOGO Puzzlers.

Only around 38% of the games surveyed incorporated some form of procedural rhetoric. Its usage tended to be more diverse and less synergistic than the narrative reinforcement strategies, with the Proteus Effect (19%) arose most commonly in immersive 3D virtual worlds where play was controlled through an avatar. The use of quantified self principles (17%) arose in RPGs, while the use of vicarious reinforcement tended to arise in simulation-style games, or those where success at the game inevitably lead to positive feedback (e.g. MUPPETS [BP04]). Although less clear to derive, there was only a single game which used process meanings to provoke reflections.

Table 7.3: A Content Analysis of Persuasive Mechanics in Programming Games

Game	Type	PE	VR	QS	PM	IE	AP	EP	AbE
APIN	[Not Available to Play]	-	-	-	-	-	-	-	-
AtlantisQuest	3D Adventure	1	0	0	0	1	1	0	1
BotLogic	2D LOGO Puzzler	0	0	0	0	0	0	0	1
Baralho das Variavies	2D Puzzler	0	0	0	0	0	0	0	1
Cargo-Bot	2D LOGO Puzzler	0	0	0	0	0	0	0	0
The Catacombs	2D RPG	1	0	1	0	1	1	0	1
Catos Hike	2D Adventure	0	0	0	0	1	1	0	1
Coddy Luck	2D Puzzler	0	0	0	0	0	0	0	0
Code Combat	2D Adventure/RPG	0	0	1	0	1	1	1	1
Code Spells	3D Sandbox Environment	1	0	0	0	1	0	0	0
Codemancer	2D LOGO Adventure/Puzzler	0	0	0	0	1	1	1	1
Colobot	3D Real-Time Strategy	1	0	1	0	1	1	0	1

Continued on next page

Table 7.3 – Continued from Previous Page

Game	**Type**	**PE**	**VR**	**QS**	**PM**	**IE**	**AP**	**EP**	**AbE**
CMX	2D Multiplayer RPG	0	0	1	0	0	0	0	0
Daisey the Dino	2D Pet Sim	0	0	0	0	0	0	0	0
Dream Coders	2D RPG	0	1	1	0	1	1	1	1
EleMental: The Recurrence	3D First-Person Puzzler	1	0	0	0	0	1	0	1
Entrando Pelo Canno	[Not Available To Play]	-	-	-	-	-	-	-	-
EraseAllKittens	2D Adventure/Platformer	0	0	0	0	0	1	0	0
Gidget	2D Sim	0	1	0	1	0	0	0	1
Hack'n'Slash	2D Adventure	0	0	0	0	1	1	0	0
IA Game	3D RPG	1	0	1	0	1	1	0	1
Java Tower Defense	2D Tower Defence	0	0	0	0	0	0	0	0
Kodable	2D LOGO Puzzler	0	0	0	0	0	0	0	0
Machineers	[Not Available to Play]	-	-	-	-	-	-	-	-

Continued on next page

Table 7.3 – Continued from Previous Page

Game	Type	PE	VR	QS	PM	IE	AP	EP	AbE
Machinist-Fabrique	[Not Available To Play]	-	-	-	-	-	-	-	-
Move the Turtle	[Not Available To Play]	-	-	-	-	-	-	-	-
Moser's Game	Text Adventure	0	0	0	0	1	1	1	1
MUPPETS	3D Multiplayer Combat Sim	0	1	0	0	0	0	0	0
Lightbot2	2D Logo Puzzler	0	0	0	0	0	0	0	1
PlayLOGO 3D	3D Logo Puzzler	0	1	0	0	0	0	0	1
Prog & Play	2D Real-Time Strategy	0	0	0	0	0	0	0	0
Program Your Robot	[Not Available To Play]	-	-	-	-	-	-	-	-
ProGame for Greenfoot	2D Sim	0	0	0	0	0	0	0	0
Project Orion	3D Adventure	1	0	0	0	1	1	0	0
RAPUNZEL	3D Dance Sim	0	1	0	0	0	0	0	0
ResourceCraft / Kernel Panic	3D Real-Time Strategy	0	1	0	0	1	0	0	0

Continued on next page

Table 7.3 – Continued from Previous Page

Game	**Type**	**PE**	**VR**	**QS**	**PM**	**IE**	**AP**	**EP**	**AbE**
RoboCom	2D Strategy	0	0	0	0	0	0	0	0
RobotLogic	[Not Available To Play]	-	-	-	-	-	-	-	-
Robotimov	[Not Available To Play]	-	-	-	-	-	-	-	-
Robozzle	2D LOGO Puzzler	0	0	0	0	0	0	0	0
Ruby Warrior	2D Rogue-like	0	1	0	0	0	0	0	1
Saving Sera	2D RPG	1	0	1	0	1	1	1	1
SpaceChem	2D LOGO Puzzler	0	0	0	0	0	1	1	0
Space Goats	2D Tower Defence	0	0	0	0	0	0	0	0
Stop Toilworm Diamond	[Not Available To Play]	-	-	-	-	-	-	-	-
TALENT	[Not Available To Play]	-	-	-	-	-	-	-	-
ToonTalk	2D Puzzle / Adventure	0	0	0	0	0	1	0	1
Train B&P	3D Train Sim	0	0	0	0	0	0	0	0

Continued on next page

Table 7.3 – Continued from Previous Page

Game	**Type**	**PE**	**VR**	**QS**	**PM**	**IE**	**AP**	**EP**	**AbE**
Tynker Lost in Space	2D LOGO Adventure/Puzzler	0	0	0	0	0	0	0	0
VirgoGame	2D LOGO Puzzler	0	0	0	0	0	0	0	0
World of Variables	2D Quiz / Sim	0	0	0	0	0	0	0	1
Wu's Castle	2D Puzzle/Adventure	0	0	0	0	1	0	0	1

Note: PE = Proteus Effect; VR = Vicarious Reinforcement; QS = Quantified Self; PM = Process Meaning; IE = Imago Effect; AP = Adventure Programming; EP = Emotion Design Patterns; AbE = Agent-based Encouragement.

These results show that a range of persuasive mechanisms exist in games; however, within those programming games included in the analysis, there is little coordinated effort to exploit the synergism between these different approaches. In particular, the use of narrative and characters appears to be more common than the use of the processes and systems. Notably, many of the most popular type of game, LOGO Puzzlers popularised by the works of Papert's [Pap80] and Bergin [BSRP05], showed little use of these persuasive mechanisms. Hence, it may be appropriate for future game designers to consider hybridising these popular approaches to teach programming with elements from other genres. For example, RPGs were the most likely type of game to combine a range of persuasive elements. Hence, incorporating a LOGO Puzzler into a broader RPG could be appropriate to maximise the affective potential of the medium alongside its apparent popularity as a method of instruction.

While persuasive mechanisms in games present an exciting opportunity, however, it is important to reflect on the lack of evidence to support their use. To illustrate, Tobias *et al* [TFDW11, p. 206] protest that "there is considerably more enthusiasm for describing the affordances of games and their motivating properties than for conducting research to demonstrate that those affordances are used to attain instructional aims [...] This would be a good time to shelve the rhetoric about games and divert those energies to conducting needed research". This assertion is supported by three recent meta-analytic reviews which presented guarded conclusions due to the diversity of dependent variables and the perceived lack of methodological rigour in many of the studies included [Pet09, Sit11, CBM+12, GEM13, WVNVOVDS13]. As such, these results should be taken as descriptive and hypothetical because there is a further need to validate the utility of the proposed mechanisms with empirical evidence, likely in the form of future experimental study.

It is also important to note that very few programming games seem to explicitly address the challenges associated with programming anxiety by targeting low self-concept or fixed programming aptitude. Yet, such play-based instructional tools potentially offer well-scaffolded preparatory material in an engaging manner and just-in-time delivery of learning material which could help students overcome a low self-concept and even help them to develop a growth mindset. As such, the conceptual framework and measurement instruments presented in the previous chapters of this thesis can be used to advance the work done in this area.

7.5 Limitations

There are two important limitations to note. Firstly, the sample of games analysed in this chapter were curated rather than sampled. Consequently, the initial list is not exhaustive in nature as additional works only available outside of the UK and unpublished works-in-progress may also exist. Additionally, those games which were analysed did not form a random sample and so may not be representative of the overall population of programming games. In particular, those not discussed in academic texts and academic mailing lists will have been omitted as a result of the curation process. Secondly, no measure of reliability has been reported alongside the content analysis conducted in this paper (see

[LSDB02], [Kri04], and [HK07] for details on reliability measurement in content analysis). This is because the games were analysed by a single rater (the author).

7.6 Conclusion

This chapter has proposed that *procedural rhetoric* and *narrative reinforcement* can be used to enrich students' self-beliefs through eight persuasive mechanisms: the Proteus Effect; Vicarious Reinforcement; Quantifiable Self; Process Meaning; Imago Effect; Agent-based Encouragement; Adventure Programming; and Emotion-orientated Design. An analysis of these mechanisms across fifty games shows that many games include at least one of these mechanisms in their design, but there is considerable variance in the way in which each is deployed and their purpose in doing so. Two key gaps in the literature are revealed. Firstly, none of the games included in the survey explicitly address the challenges associated with poor programming self-concept and fixed programming aptitude mindsets. Secondly, there is little empirical evidence associated with these games which could be used to verify the utility of the proposed mechanisms. This suggests that there is an opportunity to study the use of these persuasive mechanisms through the creation and evaluation of new experimental games that specifically targets students' self-beliefs.

Declaration

Some of the work presented in this chapter is also under review for publication in the following publication:

Scott, M. J. and Ghinea, G., Enriching the Self-Concept and Mindset of Novice Programmers using Game-based Fantasy Role-Play: A Review of Existing Games, *British Journal of Educational Technology* (Under Review), pp. 1–6, DOI: 10.1109/TE.2013.2288700

8

Exploring the 'Projective Identity' Hypothesis

It is hypothesised that the 'projective identity' which is created during a fantasy role-play could help students to develop a stronger self-concept as a programmer. Two versions of a debugging exercise were developed, with one incorporating elements of fantasy role-play. This chapter discusses the outcome of a double-blind parallel-group randomised trial which was used to test the hypothesis.

8.1 Introduction

Positive psychology claims that a reciprocal relationship exists between achievement and programming self-concept (PSC) [Hua11], defined here as "a person's self-perceptions that are formed through experience with and interpretations of one's environment" in relation to computer programming [MM11, p. 61]. Therefore, reinforcing this construct by offering well designed learnihng experiences could lead to the emergence of more effective learners.

However, it is not clear what practices could be applied in learning environments for introductory programming courses to achieve this. Due to the increasireng adoption of educational multimedia in such environments, it would seem sensible to identify how such tools can be leveraged to enhance programming self-concept effectively.

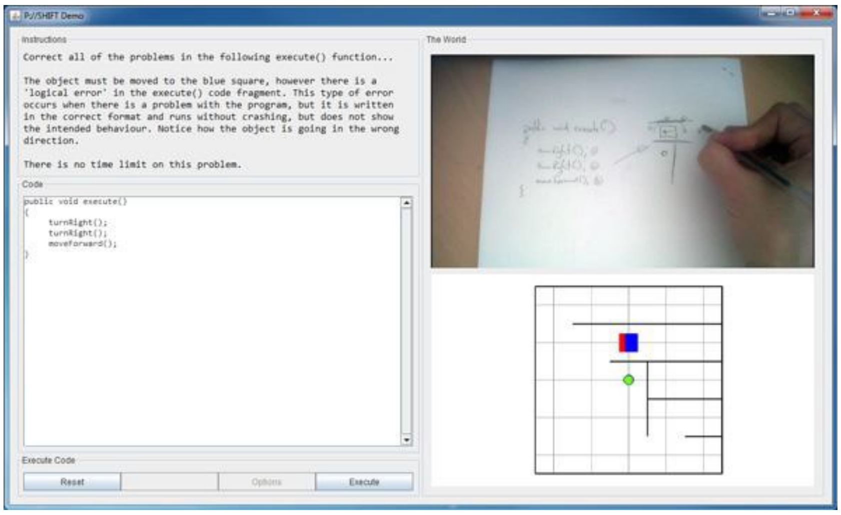
The use of fantasy role-play [Gee03, Gee14, HAB05] within a dramatic narrative [Eri96] could be one such practice. It has demonstrated some qualitative success in Space Mission: Ice Moon, enabling students to "think and act like scientists" [DG07, p.4]. Among creative learning environments [RFK$^+$09], a similar approach can also be seen in GameStar Mechanic. There, novice designers are immersed in a series of story-driven scenarios while they learn basic concepts and skills before proceeding onto more practical design activities. While the latter tool addresses games design, rather than programming, a similar approach could effectively prepare students for creative programming activities, as in [Rep12]—but conveying the preparatory material in a more timely and engaging manner to avoid a fast pace. This is because fast paces can be uncomfortable for some students, which is undesirable because of its potential impact on retention [BMK09]. Thus, to what extent could the application of fantasy role-play contribute to the development of programming self-concept?

During an exploration of learning in games, Gee [Gee03, Gee14] considered the role of identity. A three-way relationship was proposed between: real-world identity; virtual identity; and a mediating projective identity. It was argued that the virtual and projective identities formed when immersed in a fantasy role could reinforce the real-world identity relating to the activity of that role. During a separate investigation of self-representation in virtual reality, Yee and Bailenson [YB07] empirically demonstrated a somewhat similar phenomenon relating to identity and self-beliefs: the Proteus Effect. Using virtual reality headsets to immerse participants in a 3D virtual world under experimental conditions, it was shown that manipulating the avatars of participants within the virtual environment had some impact on their attitudes and behaviour. Subsequently, some of this difference was maintained when measured a short time after the experiment. However, the extensibility of this effect to other domains, such as educational multimedia, is unclear.

If the phenomenon does extend to educational multimedia, it could have significant implications for the design of learning environments and the presentation of e-learning material. Thus, this paper describes an initial experimental study, in which fantasy role-play is integrated into a prototype virtual lab exercise that aims to enhance debugging skills. It then explores the following research question: assuming an equal baseline, does the incorporation of fantasy role-play in an e-learning activity increase its impact on programming self-concept development for undergraduates enrolled on a programming course?

8.2 Tool Development

The e-learning activity was created by the author specifically for this study. Because the review conducted in the previous chapter showed that this type of activity was popular, the design drew inspiration from Logo Geometry [Pap80] and Karel the Robot [BSRP05] and aimed to increase student confidence by teaching how to identify and correct simple syntax and logic errors in snippets of Java code. Students are first shown an instructional video within the tool, explaining how to trace code to identify errors. A fragment of code containing faulty instructions to navigate a maze is then displayed. Once a student has identified and corrected the mistakes, an animation shows an object moving through the maze, revealing whether all of the errors have been removed. Advice is offered at various intervals, helping students to improve their ability in analyzing faulty code and enabling them to identify common mistakes in their own programs more readily. The interface is shown in Figure 8.1.



Correct all of the problems in the following execute() function...

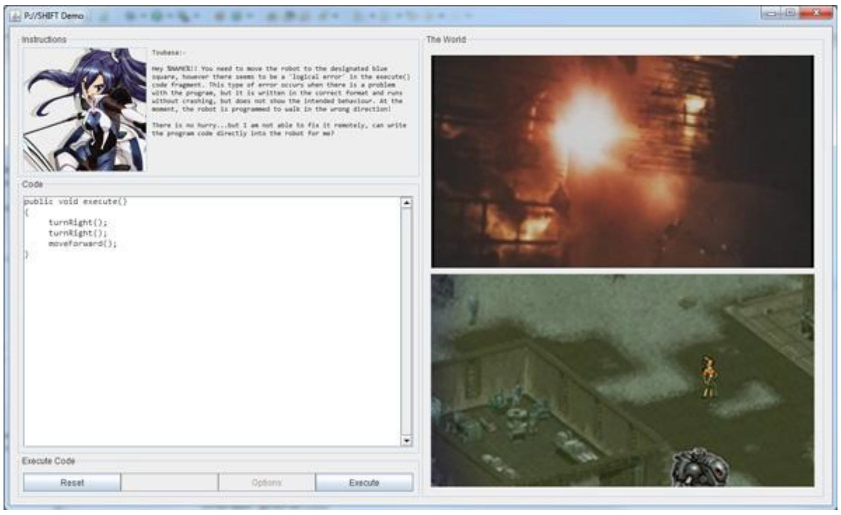
The object must be moved to the blue square, however there is a 'logical error' in the execute() code fragment. This type of error occurs when there is a problem with the program, but it is written in the correct format and runs without crashing, but does not show the intended behaviour. Notice how the object is going in the wrong direction.

There is no time limit on this problem.

```
public void execute()
{
     turnRight();
     turnRight();
     moveForward();
}
```

Figure 8.1: Learning to Trace Code.

In order to embed virtual and projective identities within the tool, elements of fantasy role-play were incorporated. In this new version, students would select an avatar, assuming the role of a computer systems specialist on an advanced interstellar spaceship. In this role, students would program repair robots to navigate the ship's maintenance areas in order to fix all of the problems. A video is shown as the tool is launched, setting the scene, and several graphical improvements were added to the user interface, as shown in Figure 8.2.



Tsubasa:-

Hey SNAKE!! You need to move the robot to the designated blue square, however there seems to be a 'logical error' in the execute() code fragment. This type of error occurs when there is a problem with the program, but it is written in the correct format and runs without crashing, but does not show the intended behaviour. At the moment, the robot is programmed to walk in the wrong direction!

There is no hurry...but I am not able to fix it remotely, can write the program code directly into the robot for me?

```
public void execute()
{
    turnRight();
    turnRight();
    moveForward();
}
```

Figure 8.2: Integrating Fantasy Role-Play.

This version of the tool is almost functionally identical to the original. The exercises are the same, requiring about 20 minutes to complete, however each problem is recast into a narrative context. This is conveyed by characters who relate the learning content, instructions and the story through dialogue. The virtual environment is also updated with new graphics, similar to those in a 2D computer game. These changes were necessary to make the exercise more immersive.

8.3 Method

A between-subjects experimental design was adopted due to the potential for preference to bias the self-belief measure. The design consisted of a parallel-group double-blind randomised trial, incorporating balanced allocation between two groups (1:1). Two versions of the virtual lab activity were compared, one incorporating fantasy role-play (experimental condition) and an ablated version, omitting elements of fantasy role-play (control condition). Scores were imputed from a self-completed measurement of academic self-beliefs towards programming, which were measured at pre-test and post-test. The difference between the two groups were compared using an ANCOVA on the post-test scores, using pre-test scores as a covariate.

8.3.1 Sample

An *a priori* power analysis in G*Power 3.1 showed that for a study of typical significance ($\alpha = .05$) attempting to detect a *large* effect ($f = .5$) [Coh92, Coh13], a sample size of 34 would be sufficient to achieve acceptable power ($1 - \beta = .80$). A sample size of 36 undergraduate computing students was obtained for this study.

The participants were recruited from a pool of students that had registered interest online (http://www.p-shift.org) in April 2012. To be eligible, participants had to be first or second year undergraduate students enrolled on programming modules at the authors' institution. Furthermore, they had to be aged 18 or over. Although some details were omitted in order to practice blinding, each participant provided informed consent.

8.3.2 Measurement

The same programming self-concept questions that were incorporated into the measurement tool developed and presented in Chapter 5 was used. No other measurement instruments were used for this experiment as the focus is on a single hypothesis.

8.3.3 Procedure

As participants registered interest in the experiment, they immediately completed the online pre-test questionnaire. Once an appropriate number had signed up, each confirmed their interest by email and they were assigned to one of two groups, of equal size, and then emailed a link to download the relevant version of the tool.

The allocation procedure was performed in Microsoft Excel 2007 (12.0 SP3). First, the identity of each participant was obfuscated using an identification number and arbitrarily allocated a row. The rows were then sorted according to a RAND() value in a separate column. Then, the worksheet was divided into two halves, representing each group.

Participants were free to complete the exercise and the online post-test questionnaire within 10 days. Email reminders were sent after a week, two days prior to the deadline, and on the day of the deadline. All participants completed the post-test questionnaire within this period. Although this resulted in less control over experimental conditions, it is more representative of how a virtual lab exercise may be used by students in practice, thus enhancing ecological validity. It should be noted, however, that no formal teaching (such as lectures, seminars, or labs) occurred during the time period of the trial. Furthermore, while the pre-test and post-test questionnaires contained the same items, the order in which they were presented was randomised.

Table 8.1: Descriptive Statistics for Each Group in the Experiment

Variable	Mean	Std Deviation	Upper CI	Lower CI
Experimental Group				
Pre-Test ASC	1.7056	0.8411	1.2873	2.1239
Post-Test ASC	1.7443	0.8303	1.3313	2.1572
Control Group				
Pre-Test ASC	1.5804	0.7807	1.2873	2.1239
Post-Test ASC	1.5991	0.7739	1.3313	2.1572

CI: Confidence Interval.

Table 8.2: Experimental Results (dv = Post-Test Programming Self-Concept)

Source of Variance	SS	df	MS	F	p	η_p^2
Pre-Test ASC	21.872	1	21.872	5.679	.000	.999
Allocation	.004	1	.004	4.181	.049	.112
Error	.032	33	44.326			

Adjusted Model R Squared = .998.

8.4 Results

The data was analyzed using PASW 18.0.3 for Windows. All cases were included in the analysis, with any missing ASC values in the raw pre-test data being replaced by the sample mean and any missing ASC values in the raw post-test data being replaced by the allocation mean.

After the allocation, one-way ANOVAs demonstrated that no significant differences existed in terms of pre-test ASC score ($F[1, 34] = .214, p = .646, \eta_p^2 = .006$) or academic grade profile ($F[1, 34] = .332, p = .568, \eta_p^2 = .010$) between the two groups. Descriptive statistics are shown in Table 8.1. An ANCOVA examined the impact of fantasy role-play on developing programming self-concept. Assumptions of normality, homogeneity of variance and homogeneity

Table 8.3: Estimated Marginal Means (*dv* = Post-Test Programming Self-Concept)

Group	Mean	Std Error	Lower CI	Upper CI
Experimental Group	1.682	.007	1.667	1.698
Control Group	1.661	.007	1.646	1.676

CI: Confidence Interval; Covariates: Pre-Test ASC = 1.643.

of regression were verified. The results are summarized in Table 8.2. Hence, after controlling for pre-test scores, there was a significant difference ($p < .05$) in the post-test ASC between the two groups, which constitutes a *medium-to-large* effect size [Coh92, Coh13]. This difference can be seen more clearly in the estimated marginal means, shown in Table 8.3. Based on the means presented above (and assuming a baseline score of 1.643), the gain in programming self-concept for those allocated to the fantasy role-play condition (2.4%) was greater than those in the control condition (1.1%).

8.5 Discussion

Although a significant difference was detected ($p < .05$), the effect was lower than predicted, resulting in low power ($1 - \beta = .54$). This effect, estimated from the η_p^2 value, being $f = 0.355$, and, estimated from the observed difference in gain scores, being $d = 0.619$. These results support the notion that fantasy role-play can enhance the development of self-concept, as shown by the gain scores in Figure 8.3.

However, the impact had no direct practical relevance in that the small-scale isolated exercise, in isolation, was not sufficient to produce a meaningful change in

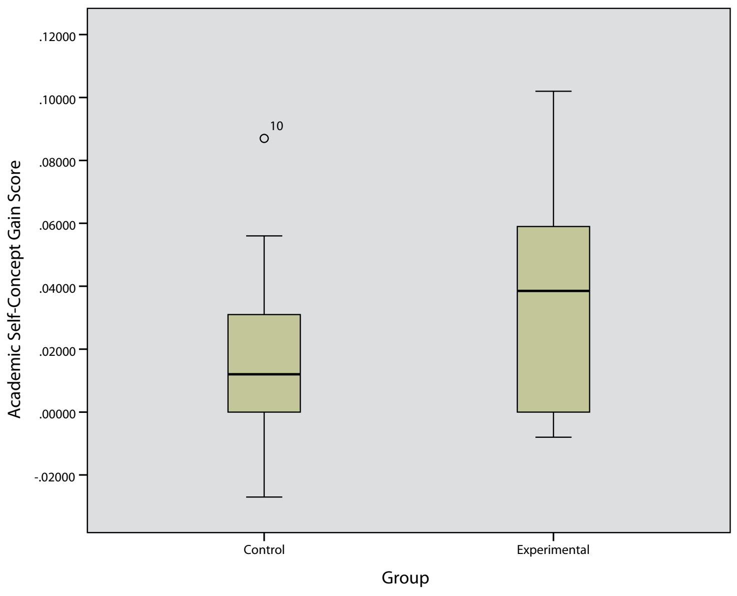


Figure 8.3: A Box-Whisker Plot Illustrating the Gain Scores Within Each Experimental Allocation.

programming self-concept. Nevertheless, if a small effect is present and it can be sustained, then it may result in a more practically significant gain over multiple tasks. Thus, a longitudinal trial is proposed as future work. Assuming that the observed gain remains consistent ($f = .031$), then perhaps eight activities could produce a more meaningful change ($f \approx .253$).

8.6 Conclusion

A prototype of a debugging exercise that incorporates elements of fantasy role-play has been shown to strengthen programming self-concept to a greater extent than a conventional approach. This supports the hypothesis that projective identities enhance the self-concept development of novice programmers on a computing course. However, the difference was only of *medium* size and neither condition had a practically significant impact. Due to the limited scope of this initial study, it remains unclear whether such a modest gain could be further enhanced over time and then maintained. Thus, additional longitudinal study is appropriate. Moreover, qualitative enquiry is necessary to reveal how the fantasy role-play could be improved for greater impact. Future research should anticipate *medium* effects [Coh92, Coh13].

Declaration

Some of the work presented in this chapter can also be found in the following publication:

Scott, M. J. and Ghinea, G., Integrating narrative reinforcement into the Programming Lab: Exploring the 'Projective Identity' Hypothesis, *Proceedings of 43rd ACM Technical Symposium on Computer Science Education (Denver, CO, USA)*, March 2013, pp. 119–122

9

Conclusion

This final chapter of the thesis re-examines the key findings from the research programme and emphasises the key contributions made to the field of computing education. A short retrospective addressing the limitations of this programme is presented. Then the implications for theory and practice are described, leading into a brief discussion on avenues for further research.

9.1 Overview

It can be notoriously difficult to teach computer programming to undergraduate students. This is, to some extent, because many students do not develop strong beliefs in themselves and thereby do not readily engage in self-regulated deliberate practice. Thus, close support and encouragement is needed to help maintain students' level of practice. Such support, however, is not always scalable and so there is a need for alternative formats which are more scalable. Educational multimedia presents itself as one such viable format; however, in order to proceed with the design and creation of new multimedia-based support tools, a clear understanding of student practice and the relative affordances of multimedia technology is needed.

Games have been evangelised as a source of innovation in education, with a range of hypothesised applications and benefits including the enrichment of self-beliefs. Thus, the research presented in this doctoral thesis aimed to address the challenge by, firstly, extending our current understanding of the self-beliefs of novice programmers and then by, secondly, investigating the impact of using games-based fantasy role-play to enrich these self-beliefs. The work was, broadly speaking, conceptually modelled using the Control Value Theory of Achievement Emotions and focused on four key constructs derived from the model: programming self-concept; programming aptitude mindset; programming anxiety; and amount of programming practice.

An initial challenge with this work was the maturity of the computing education research literature compared to similar fields such as physics and mathematics. Notably, from the initial literature review, it was not clear whether or not a domain-focused or general educational theory should be applied. Therefore, the first study questioned whether a domain-specific approach would be needed by comparing a self-belief situated in the programming domain with a self-belief in the intelligence domain. This work revealed that the domain-specific approach had greater predictive utility.

However, this led into another challenge. It became clear that a domain-specific measurement instruments would be needed for a programming-specific approach to self-belief research. As such, the second study proposed a novel measurement instrument based on items drawn from several pre-existing instruments. These modifications were adapted and re-validated in the context of three cohorts of programming students, revealing adequate psychometric support.

Examination of the relationships provided empirical support for the basic elements of the Control Value Theory of Achievement Emotions. Notably, that higher anxiety lowered practice. However, of some concern, was that a comparison between the web programming course ran in 2011-12 and the robot-centred course ran in 2012-13 cohort did not reveal any significant differences in terms of self-concept or anxiety. Additionally, amount of practice was shown not to be the sole element of programming success, revealing no significant relationship between code readability or code sophistication.

Given that no significant differences were found, the potential of games-based fantasy role-play presents itself as a compelling alternative (or complement) to the use of personal robots in a robot-centred introductory course. However, despite the promise of games, the claims made by many serious game evangelists, there is limited empirical evidence. In particular, there are few studies which empirically explore the impact of games on students' academic beliefs. To see if this was the case within the computing education research literature, a review and detailed analysis of programming games was conducted. This revealed that, indeed, there was little empirical evaluation of the effect of games on academic beliefs in the programming context. Additionally, the programming games curated for the review seldom integrated key features for enhancing such beliefs.

The final experiment presented in the thesis aimed to address this lack of integration of features for enhancing student self-beliefs into programming games.

However, as a result of the lack of existing programming games with suitable qualities, a new programming game embedding some of the principles associated with the enrichment of self-concept was implemented. An alternative version for comparison was also created. The results of a randomised trial showed a modest, but statistically significant, impact on programming self-concept. Thus, promising that games-based fantasy role-play can be used as a means to improve student self-beliefs in the programming context. However, that its effects should not be overestimated.

9.2 Key Contributions

Several contributions to discourse in computing education have been made as a result of the work carried out in this thesis. Each claim refers to its respective chapter for ease of reference and is listed as follows:

> ***Claim IV-I:*** *It is important to use domain contexts and domain-specific measurement in self-belief research.*
>
> ***Claim V-I:*** *The Self-Belief Enrichment Questionnaire offers a valid approach to measurement in programming self-belief research.*
>
> ***Claim V-II:*** *The Control-Value Theory of Achievement Emotion is an appropriate conceptual framework for the investigation of self-beliefs and experiences of novice programmers.*
>
> ***Claim VI-I:*** *Robot-centred programming courses demonstrate no benefit over web programming courses in terms of programming self-concept and programming anxiety.*

Claim VI-II: *Instructional strategies centred upon robots can be used to enrich both the quality and the quantity of programming practice.*

Claim VIII-I: *Instructional strategies that incorporate games-based fantasy role-play will (modestly) enhance programming self-concept.*

9.3 Implications for Practice

The claims derived from the work presented in this thesis suggest several implications for educational practice. The study presented in Chapter 4 shows that a domain-specific mindset construct is a better predictor of programming practice compared to a domain-general construct. As such, educational interventions that target more general constructs could have a sub-optimal impact. Therefore, such interventions should include domain-specific elements. For example, when providing feedback to students, it is important to tailor such feedback to include programming-specific aspects. The study presented in Chapter 5 shows that some aspects of the Control-Value Theory of Achievement Emotions are relevant within the programming education context. In particular, that programming anxiety has an inhibiting influence on time spent practising programming. As such, it is important to minimise programming anxiety. The study also shows that programming self-concept and programming aptitude mindset are correlated with such anxiety. Consequently, the application of self-concept and mindset interventions is to be encouraged. Some methods for achieving this include providing motivating feedback (as described by [CCD+10]) alongside other strategies such as soft-scaffolding to ensure challenges remain in a students' zone of proximal development [SK07b] while designing the learning experience so it evokes a feeling of pride [GCF+10]. The study presented in Chapter 6 shows that practice can still be improved in spite of the challenges associated with anxiety. The students using personal robots practiced more and wrote more sophisticated code. This suggests that an engaging learning context alongside a immediate visual feedback can improve student engagement as well as student outcomes. One such environment which can provide this is game-based fantasy role-play. However, the small effect size found in the trial of the tool presented in Chapter 8 suggests that these types of tool may need to be used consistently across a course rather than as a one-off learning activity.

9.4 Implications for the Field

The work presented in this thesis clearly outlines a need for validated domain-specific research instruments. This is because the work presented in Chapter 4 demonstrates that domain-specific construct can have stronger relationships with key variables of interest than a domain-general construct. While this does not invalidate previous findings that did not use validated instruments (e.g., the trend found in [MD04]), the interpretation of individual results must be made with care, especially where the effect sizes are small. This is because it may not be clear what is being measured, the reliability of such instruments may not have been assured, and the effect size may be an under-estimation compared to its domain-specific analogue. It is, therefore, hoped that the recommendations made by Tew and Dorn [TD13] are adopted and that researchers in the field assess their research instruments in line with the method presented in Chapter 5.

The study presented in Chapter 5 demonstrates that the Control-Value Theory of Achievement Emotion is an appropriate conceptual framework for the investigation of programming anxiety. However, there are many opportunities to build on this work in the programming context; both, in terms of studying a wider range of emotions and in terms of studying a wider range of antecedents. Additionally, the existing model based on self-concept and mindset can be applied to different range of problems (i.e., different dependent variables) by extending the CS1 self-belief questionnaire.

The final experiment presented in Chapter 8 suggests further work is needed on improving the design and deployment of games-based fantasy role-plays in the programming context. However, the analysis of games presented in Chapter 7 raises some concern about the small number of programming games which have been rigorously evaluated; or even just designed to help overcome some of challenges faced by programming students. While there is evidence for the efficacy of such games for learning and motivation in related areas **??**, it would be pragmatic for practitioners in the field to focus their designs and evaluation on the specific challenges faced by students in introductory programming. For example, the development of non-constructive self-beliefs (as highlighted in Chapter 4) and their subsequent impact on learning behaviours (as implied in Chapter 5).

9.5 Limitations

In retrospect, there are a range of limitations to this research. The use of a convenience sample during the early stages of the research (notably, in Chapter 4) has resulted in lost opportunities for further insight. While this does not threaten the overall validity of the findings, it limited the exploration to relationships between variables and restricted such exploration to only between those variables with a full range of responses. As such, interesting questions of a demographic nature, such as the proportion of fixed mindsets in a typical student cohort had to be excluded.

Throughout the research, a self-report measure of programming practice has been used. This was required because no systematic approach to measuring was available at the institution. Caution must be advised when interpreting such measures because of the potential for biases (e.q. acquiescence bias) [DGV02a].

Statistical power has been a concern throughout the work because of the challenges associated with recruiting a large and representative sample of introductory programming students. Of particular note is the proportion of female students in the cohort which compromised the power of test between certain variables (i.e., the effect sizes for differences in gender were as high as 0.3 in Chapter 6 but were non-significant, potentially being Type-II errors). Additionally, involving students in complex experiences, such as the trial described in Chapter 8, was particularly difficult. Only a small sample of 36 students was obtained. This resulted in insufficient power to warrant a timely replication attempt.

Additionally the focus of this research was quantitative methods, excluding opportunities to gather potentially useful insight on the issues of self-beliefs. This is largely due to the author's lack of previous experience using qualitative methods and significant investment in time which was required to learn educational and measurement theory. During the initial planning of the research programme, this was justified on the basis that a lot of qualitative work is already being conducted in the area of self-beliefs and student emotions within the computing education research literature. However, there is little in the context of designing programming games. Participatory design would likely improve the experimental game presented in Chapter 8.

9.6 Future Directions

In order to overcome some of the limitations of the research, there are several future directions. Firstly, there is a need to conduct a longitudinal study on the impact of games-based fantasy role-play, covering a broader range of self-belief constructs. This is because it is not clear whether there will be interaction effects, whether the modest observed effect will stack, whether it is the result of acquiescence bias or a novelty effect, or whether there may be a habituation effect which arises after a certain period of time. Secondly, the incorporation of electronic worksheets and automatic marking systems into future studies (e.g., [SFSR13, SDR+15]) would enable improved measurement of programming practice. This would provide several advantages over a self-report measure, as it would provide breakdowns of programming habits that are free of acquiescence bias and could be adapted to provide insight into the quality of student practice. Thirdly, additional work is needed to support the cross-cultural validity of the self-belief measurement instrument by trialling it in other institutions. Such trails could be tested for factorial validity and invariance, thereby eliciting support for the tool to be used by a broader population. Fourthly, additional qualitative work would be useful in shaping the design of future game-based fantasy role-playing scenarios. Such participatory designs and human-centred approach would consequently help the experimental prototype to advance beyond the laboratory and into a usable product.

Further studies could also be conducted to extend the conceptual framework. Most notably, variables associated with the value component of the theory could be introduced in order to improve predictive utility. Furthermore, only anxiety was considered in this thesis so a broader range of achievement and learning emotions could be added. Further improvements could include integration with core self-evaluation theory. This is because many of its constructs serve as antecedents (and formative correlates) to self-concept and their inclusion would be useful to help monitor the impacts of interventions in their respective aspects. Additional variables could also be included in the framework in order to support research into gender, racial, and cultural differences in line with recent efforts to broaden participation (e.g., [CHH+11, GEME14, RRZMndM15]).

References

[Abt87] Abt, C. C., Serious games, University Press of America, 1987. 108

[AL13] Ahadi, A. and Lister, R., Geek Genes, Prior Knowledge, Stumbling Points and Learning Edge Momentum: Parts of the One Elephant?, *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research (New York, NY, USA)*, 2013, pp. 123–128. 13

[Ali13] Alimisis, D., Educational Robotics: Open Questions and New Challenges, *Themes in Science & Technology Education*, 6 (2013), pp. 67–71. 87

[ALM+10] Adams, D. B., Louis, R., Morin, B., Cerrato, J., Keidel, J., Vincent, J., Merrill, J., Rampelli, D., Gieskes, K., Fellows, S., et al., Explore-create-present: A project series for CS, *Proceedings of the ASEE North Central Sectional Conference (ASEE10)*, 2010, pp. 2B.1–2B.5. 86

[ALP10] Apiola, M., Lattu, M., and Pasanen, T. A., Creativity and intrinsic motivation in computer science education: experimenting with robots, *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, 2010, pp. 199–203. 87

[AMM+12] Adams, D. M., Mayer, R. E., MacNamara, A., Koenig, A., and Wainess, R., Narrative games for learning: Testing the discovery and narrative hypotheses, *Journal of Educational Psychology*, 104 (2012), p. 235. 109

[AS12] Anderson, T. and Shattuck, J., Design-Based Research A Decade of Progress in Education Research?, *Educational researcher*, 41 (2012), pp. 16–25. 27

[BA02] Burnham, K. P. and Anderson, D. R., Model selection and multimodel inference: a practical information-theoretic approach, Springer, 2002. 60

[Ban77] Bandura, A., Self-efficacy: Toward a unifying theory of behavioral change, *Psychological Review*, 84 (1977), pp. 191 – 215. 69

[BB03] Biesta, G. and Burbules, N. C., Pragmatism and educational research, Rowman & Littlefield Lanham, MD, 2003. 32, 33

[BC07] Bennedsen, J. and Caspersen, M. E., Failure rates in introductory programming, *SIGCSE Bulletin*, 39 (2007), pp. 32 – 36. 11

[BDS08] Bornat, R., Dehnadi, S., and Simon, Mental models, consistency and programming aptitude, *Proceedings of the tenth conference on Australasian computing education-Volume 78*, 2008, pp. 53–61. 14

[BED+03] Barros, J. P., Estevens, L., Dias, R., Pais, R., and Soeiro, E., Using lab exams to ensure programming practice in an introductory programming course, *ACM SIGCSE Bulletin*, 35 (2003), pp. 16–20. 91

[Bee02] Beer, J. S., Implicit self-theories of shyness, *Journal of Personality and Social Psychology*, 83 (2002), pp. 1009–1024. 48

[BH95] Benjamini, Y. and Hochberg, Y., Controlling the false discovery rate: a practical and powerful approach to multiple testing, *Journal of the Royal Statistical Society. Series B (Methodological)*, 57 (1995), pp. 289–300. 56

[BKH01] Bartlett, J. E., Kotrlik, J. W., and Higgins, C. C., Organizational Research: Determining Appropriate Sample Size in Survey Research, *Information Technology, Learning, and Performance*, 19 (2001), pp. 43 – 50. 73

[BL01] Byrne, P. and Lyons, G., The effect of student attributes on success in programming, *ACM SIGCSE Bulletin*, 33 (2001), pp. 49–52. 13

[BLR+13] Beavers, A. S., Lounsbury, J. W., Richards, J. K., Huck, S. W., Skolits, G. J., and Esquivel, S. L., Practical considerations for using exploratory factor analysis in educational research, *Practical Assessment, Research & Evaluation*, 18 (2013), p. 2. 54

[BM05] Beaubouef, T. and Mason, J., Why the High Attrition Rate for Computer Science Students: Some Thoughts and Observations, *SIGCSE Bulletin*, 37 (2005), pp. 103 – 106. 2, 11

[BMK09] Barker, L. J., McDowell, C., and Kalahar, K., Exploring Factors That Influence Computer Science Introductory Course Students to Persist in the Major, *SIGCSE Bull.*, 41 (2009), pp. 153–157, DOI: 10.1145/1539024.1508923. 130

[Bog07] Bogost, I., Persuasive games: The expressive power of videogames, Mit Press, 2007. 111, 114

[Bor11] Bornat, R., Some problems of teaching (learning) first-year programming (plus a glimmer of hope), Available from: http://www-new1.heacademy.ac.uk/assets/Documents/subjects/ics/11th_some%20_problems_teaching_first-year_programming.pdf, 2011. 2

[Bor14] Bornat, R., Camels and humps: a retraction, Available from: http://eis.sla.mdx.ac.uk/staffpages/r_bornat/papers/camel_hump_retraction.pdf, 2014. 15

[Bou86] Boulay, B. D., Some difficulties of learning to program, *Journal of Educational Computing Research*, 2 (1986), pp. 57–73. 14

[Boy90] Boyer, E. L., Scholarship Reconsidered: Priorities of the Professoriate, Carnegie Foundation for the Advancement of Teaching, NJ, USA, 1990. 27, 28

[BP04] Bierre, K. J. and Phelps, A. M., The use of MUPPETS in an introductory java programming course, *Proceedings of the 5th conference on Information technology education*, 2004, pp. 122–127. 119

[BS03] Bong, M. and Skaalvik, E., Academic Self-Concept and Self-Efficacy: How Different Are They Really?, *Educational Psychology Review*, 15 (2003), pp. 1 – 40. 69

[BSCH14] Brown, N. C., Sentance, S., Crick, T., and Humphreys, S., Restart: The resurgence of computer science in UK schools, *ACM Transactions on Computing Education (TOCE)*, 14 (2014), p. 9. 10

[BSRP05] Bergin, J., Stehlik, M., Roberts, J., and Pattis, R., Karel J Robot: A gentle introduction to the art of object-oriented programming in Java, Dream Songs Press, 2005. 125, 131

[BT95] Barr, R. B. and Tagg, J., From teaching to learning—A new paradigm for undergraduate education, *Change: The magazine of higher learning*, 27 (1995), pp. 12–26. 2

[BTD07] Blackwell, L., Trzesniewski, K., and Dweck, C. S., Implicit Theories of Intelligence Predict Achievement Across an Adolescent Transition: A Longitudinal Study and an Intervention, *Child Development*, 78 (2007), pp. 246 – 263. 67

[Car06a] Carlsson, S. A., Towards an information systems design research framework: A critical realist perspective, *Proceedings of the First International Conference on Design Science Research in Information Systems and Technology (Claremont, CA)*, 2006, pp. 192–212. 26

[Car06b] Carter, L., Why students with an apparent aptitude for computer science don't choose to major in computer science, *ACM SIGCSE Bulletin*, 38 (2006), pp. 27–31. 11

[Cas08] Castronova, E., Synthetic worlds: The business and culture of online games, University of Chicago press, 2008. 108

[CB07] Caspersen, M. E. and Bennedsen, J., Instructional design of a programming course: a learning theoretic approach, *Proceedings of the third international workshop on Computing education research*, 2007, pp. 111–122. 49

[CBM+12] Connolly, T. M., Boyle, E. A., MacArthur, E., Hainey, T., and Boyle, J. M., A systematic literature review of empirical evidence on computer games and serious games, *Computers & Education*, 59 (2012), pp. 661–686. 109, 116, 125

[CCD+10] Cutts, Q., Cutts, E., Draper, S., O'Donnell, P., and Saffrey, P., Manipulating mindset to positively influence introductory programming performance, *Proceedings of the 41st ACM technical symposium on Computer science education*, 2010, pp. 431–435. 21, 49, 51, 63, 64, 67, 107, 144

[CDP00] Cooper, S., Dann, W., and Pausch, R., Alice: a 3-D tool for introductory programming concepts, *Journal of Computing Sciences in Colleges*, 15 (2000), pp. 107–116. 116

[Cha13] Chalmers, A. F., What is this thing called science?, Hackett Publishing, 2013. 28

[CHH+11] Crutchfield, O. S. L., Harrison, C. D., Haas, G., Garcia, D. D., Humphreys, S. M., Lewis, C. M., and Khooshabeh, P., Berkeley Foundation for Opportunities in Information Technology: A Decade of Broadening Participation, *ACM Transactions on Computing Education*, 11 (2011), pp. 15:1–15:24, DOI: 10.1145/2037276.2037279. 149

[CL96] Cordova, D. I. and Lepper, M. R., Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice, *Journal of educational psychology*, 88 (1996), p. 715. 108

[CLB07] Caspersen, M. E., Larsen, K. D., and Bennedsen, J., Mental models and programming aptitude, *ACM SIGCSE Bulletin*, 39 (2007), pp. 206–210. 13

[Cle01] Clear, T., Research paradigms and the nature of meaning and truth, *ACM SIGCSE Bulletin*, 33 (2001), pp. 9–10. 31

[Cle13] Clear, T., Doctoral work in computing education research: beyond experimental designs, *ACM Inroads*, 4 (2013), pp. 28–30. xi, 36

[Coh92] Cohen, J., A power primer, *Psychological bulletin*, 112 (1992), p. 155. 61, 134, 137, 139

[Coh13] Cohen, J., Statistical power analysis for the behavioral sciences (Revised Edition), Academic press, 2013. 134, 137, 139

[Cou12] Court, S., UK Universities and College Union, 2012. 22, 107

[CSH07] Connolly, T. M., Stansfield, M., and Hainey, T., An application of games-based learning within software engineering, *British Journal of Educational Technology*, 38 (2007), pp. 416–428. 109

[CSH08] Connolly, T., Stansfield, M., and Hainey, T., Development of a general framework for evaluating games-based learning, *Proceedings of the 2nd European conference on games-based learning*, 2008, pp. 105–114. 117

[CV13] Cappelleri, D. and Vitoroulis, N., The Robotic Decathlon: Project-Based Learning Labs and Curriculum Design for an Introductory Robotics Course, *Education, IEEE Transactions on*, 56 (2013), pp. 73–81, DOI: 10.1109/TE.2012.2215329. 88

[D$^+$94] Dempsey, J. V. et al., Instructional Gaming: Implications for Instructional Technology, Available from: http://eric.ed.gov/?id=ED368345, 1994. 108

[DB06] Dehnadi, S. and Bornat, R., The camel has two humps, Available from: http://www.eis.mdx.ac.uk/research/PhD Area/saeed/paper1.pdf, 2006. 15

[DD78] Diener, C. and Dweck, C., An analysis of learned helplessness: Continuous changes in performance, strategy, and achievement cognitions following failure, *Journal of Personality and Social Psychology*, 36 (1978), pp. 451–462. 48, 51, 62

[DeV12] DeVellis, R. F., Scale Development: Theory and Applications, 3rd ed., Sage: London, 2012. 66, 71, 81

[DF06] De Freitas, S. I., Using games and simulations for supporting learning, *Learning, media and technology*, 31 (2006), pp. 343–358. 108

[DG07] Daanen, H. and Grant, L., Space Mission: Ice Moon, *ACM SIGGRAPH 2007 educators program*, 2007, p. 19. 115, 130

[DGV02a] Donaldson, S. I. and Grant-Vallone, E. J., Understanding self-report bias in organizational behavior research, *Journal of Business and Psychology*, 17 (2002), pp. 245–260. 55, 147

[DGV02b] Donaldson, S. I. and Grant-Vallone, E. J., Understanding self-report bias in organizational behavior research, *Journal of Business and Psychology*, 17 (2002), pp. 245–260. 104

[Dij89] Dijkstra, E. W., A debate on teaching computer science: on the cruelty of really teaching computer science, *Communications of the ACM*, 32 (1989), pp. 1398–1404. 14, 49

[DM08] Dweck, C. S. and Master, A., Self-theories motivate self-regulated learning, *Motivation and self-regulated learning: Theory, research, and applications*, 2008, pp. 31–51. 48, 67

[Don07] Dondlinger, M. J., Educational video game design: A review of the literature, *Journal of applied educational technology*, 4 (2007), pp. 21–31. 108

[DSN$^+$11] Deterding, S., Sicart, M., Nacke, L., O'Hara, K., and Dixon, D., Gamification. using game-design elements in non-gaming contexts, *CHI'11 Extended Abstracts on Human Factors in Computing Systems*, 2011, pp. 2425–2428. 108

[DT13] Dorn, B. and Tew, A. E., Becoming experts: Measuring attitude development in introductory computer science, *Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13)*, 2013, pp. 183 – 188. 67

[Dwe99] Dweck, C. S., Self-Theories: Their Role in Motivation, Personality, and Development, Psychology Press, Philadelphia, PA, 1999. ix, 16, 17, 20, 48, 54, 67, 70

[DWN13] Dormann, C., Whitson, J. R., and Neuvians, M., Once More With Feeling Game Design Patterns for Learning in the Affective Domain, *Games and Culture*, 6 (2013), pp. 215–237. 112, 115

[EAK08] Erodogan, Y., Aydin, E., and Kabaca, T., Exploring the psychological predictors of programming achievement, *Journal of Instructional Psychology*, 35 (2008), pp. 264–270. 13

[EFG13] Esper, S., Foster, S. R., and Griswold, W. G., CodeSpells: embodying the metaphor of wizardry for programming, *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, 2013, pp. 249–254. 116

[EKTR93] Ericsson, K. A., Krampe, R., and Tesch-Römer, C., The Role of Deliberate Practice in the Acquisition of Expert Performance, *Psychological Review*, 100 (1993), pp. 363–394. 2, 11, 13, 62, 102

[EN07] Egenfeldt-Nielsen, S., Third generation educational use of computer games, *Journal of Educational Multimedia and Hypermedia*, 16 (2007), pp. 263–281. 108

[Eri96] Erion, P., Drama in the classroom: creative activities for teachers, parents & friends, Lost Coast Press, 1996. 130

[EW95] Eccles, J. S. and Wigeld, A., In the mind of the actor: The structure of adolescents' achievement task values and expectancy-related beliefs, *Personality and Social Psychology Bulletin*, 21 (1995), pp. 215–225. 69

[FB09] Fox, J. and Bailenson, J. N., Virtual self-modeling: The effects of vicarious reinforcement and identification on exercise behaviors, *Media Psychology*, 12 (2009), pp. 1–25. 111, 113

[FCSC10] Freyne, J., Coyle, L., Smyth, B., and Cunningham, P., Relative status of journal and conference publications in computer science, *Communications of the ACM*, 53 (2010), pp. 124–132. 31

[fE13] for Education, D., National curriculum from September 2014, 10

[FL81] Fornell, C. and Larcker, D. F., Evaluating structural equation models with unobservable variables and measurement error, *Journal of marketing research*, 18 (1981), pp. 39–50. 78

[FM02] Fagin, B. S. and Merkle, L., Quantitative Analysis of the Effects of Robots on Introductory Computer Science Education, *ACM Journal of Educational Resources in Computing*, 2 (2002), pp. 1–18. 87, 88

[FO05] Froyd, J. E. and Ohland, M. W., Integrated engineering curricula, *Journal of Engineering Education*, 94 (2005), pp. 147–164. 92

[Fro07] Frome, J., Eight Ways Videogames Generate Emotion, *Proceedings of the 2007 DiGRA International Conference: Situated Play*, September 2007, pp. 831–835. 112, 115

[FT06] Fletcher, J. and Tobias, S., Using computer games and simulations for instruction: A research review, *Proceedings of the Society for Advanced Learning Technology Meeting*, 2006. 108

[Fur12] Furber, S., Shut Down or Restart? The Way Forward for Computing in UK Schools, The Royal Society, 2012. 10

[FVC09] Ferla, J., Valcke, M., and Cai, Y., Academic Self-Efficacy and Academic Self-Concept: Reconsidering Structural Relationships, *Learning and Individual Differences*, 19 (2009), pp. 499 – 505. 49, 69

[Gar06] Gardner, H., Multiple intelligences: New horizons, Basic Books, 2006. 48

[GC02] Gürer, D. and Camp, T., An ACM-W Literature Review on Women in Computing, *SIGCSE Bull.*, 34 (2002), pp. 121–127, DOI: 10.1145/543812.543844. 90

[GCF$^+$10] Goetz, T., Cronjaeger, H., Frenzel, A. C., Lüdtke, O., and Hall, N. C., Academic self-concept and emotion relations: Domain specificity and age effects, *Contemporary Educational Psychology*, 35 (2010), pp. 44–58. 18, 21, 107, 115, 145

[Gee03] Gee, J. P., What video games have to teach us about learning and literacy, *Computers in Entertainment (CIE)*, 1 (2003), pp. 20–20. 109, 113, 116, 130

[Gee14] Gee, J. P., What video games have to teach us about learning and literacy, Macmillan, 2014. 109, 112, 114, 116, 130

[Gei94] Geitz, R., Concepts in the classroom, programming in the lab, *ACM SIGCSE Bulletin*, 26 (1994), pp. 164–168. 91

[GEM13] Girard, C., Ecalle, J., and Magnan, A., Serious games as new educational tools: how effective are they? A meta-analysis of recent studies, *Journal of Computer Assisted Learning*, 29 (2013), pp. 207–219. 109, 125

[GEME14] Guzdial, M., Ericson, B., Mcklin, T., and Engelman, S., Georgia Computes! An Intervention in a US State, with Formal and Informal Education in a Policy Context, *ACM Transactions on Computing Education*, 14 (2014), pp. 13:1–13:29, DOI: 10.1145/2602488. 149

[GL$^+$94] Guba, E. G., Lincoln, Y. S., et al., Competing paradigms in qualitative research, *Handbook of qualitative research*, 2 (1994), pp. 163–194. 31

[GMB03] Guay, F., Marsh, H. W., and Boivin, M., Academic self-concept and academic achievement: Developmental perspective on their causal ordering, *Journal of Educational Psychology*, 95 (2003), pp. 124 – 136. 70

[Gov12] Gove, M., Digital Literacy and the Future of ICT in Schools Presentation at the British Educational Training and Technology (BETT) Show, Available from: http://www.education.gov.uk/inthenews/speeches/a00201868/michael-gove-speech-at-the-bett-show-2012, June 2012. 10

[GRD12] Good, C., Rattan, A., and Dweck, C. S., Why do women opt out? Sense of belonging and women's representation in mathematics, *Journal of personality and social psychology*, 102 (2012), pp. 700–717. 48

[Gre06] Gregor, S., The nature of theory in information systems, *MIS Quarterly*, 30 (2006), pp. 611–642. 35

[Gre09] Gregor, S., Building Theory in the Sciences of the Artificial, *Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology (New York, NY, USA)*, 2009, pp. 4:1–4:10. 35

[GS90] Gigerenzer, G. and Swijtink, Z., The empire of chance: How probability changed science and everyday life, vol. 12, Cambridge University Press, 1990. 42

[GS04] Gibbs, G. and Simpson, C., Conditions under which assessment supports students learning, *Learning and teaching in higher education*, 1 (2004), pp. 3–31. 16, 91

[GS12] Gaydos, M. J. and Squire, K. D., Role playing games for scientific citizenship, *Cultural Studies of Science Education*, 7 (2012), pp. 821–844. 108

[Guz11] Guzdial, M., From science to engineering: Exploring the Dual Nature of Computing Education Research, *Communications of the ACM*, 54 (2011), pp. 37–39. 2, 11

[HAB05] Habgood, M., Ainsworth, S., and Benford, S., Endogenous fantasy and learning in digital games, *Simulation & Gaming*, 36 (2005), pp. 483–498. 108, 130

[Han00] Hans, T. A., A meta-analysis of the effects of adventure programming on locus of control, *Journal of contemporary psychotherapy*, 30 (2000), pp. 33–60. 112, 115

[Hat09] Hattie, J., Visible learning: A synthesis of over 800 meta-analyses relating to achievement, Routledge, 2009. 12

[Hay05] Hays, R. T., The effectiveness of instructional games: A literature review and discussion, Tech. report, DTIC Document, 2005. 108

[HBBA10] Hair, J., Black, W., Babin, B., and Anderson, R., Multivariate Data Analysis, Seventh Edition, Psychology Press, NJ, USA, 2010. xi, 56, 57, 72, 73, 75, 77, 78, 81

[HCSB11] Hainey, T., Connolly, T. M., Stansfield, M., and Boyle, E. A., Evaluation of a game to teach requirements collection and analysis in software engineering at tertiary education level, *Computers & Education*, 56 (2011), pp. 21–35. 108, 109

[HH$^+$11] Hilton, M., Honey, M. A., et al., Learning science through computer games and simulations, National Academies Press, 2011. 108

[HK07] Hayes, A. F. and Krippendorff, K., Answering the call for a standard reliability measure for coding data, *Communication methods and measures*, 1 (2007), pp. 77–89. 95, 127

[HR97] Heron, J. and Reason, P., A participatory inquiry paradigm, *Qualitative inquiry*, 3 (1997), pp. 274–294. 32, 34, 35

[Hua11] Huang, C., Self-Concept and Academic Achievement: A Meta-Analysis of Longitudinal Relations, *Journal of School Psychology*, 49 (2011), pp. 505 – 528. 18, 129

[Hug04] Huggard, M., Programming Trauma: Can it be Avoided?, *Paper presented at the BCS Conference on Grand Challenges in Computing: Education*, 2004, p. 50. 2, 12, 67

[Hui86] Huizinga, J., Homo Ludens Ils 86, Routledge, 1986. 108

[JB01] Judge, T. A. and Bono, J. E., Relationship of core self-evaluations traitsself-esteem, generalized self-efficacy, locus of control, and emotional stabilitywith job satisfaction and job performance: A meta-analysis, *Journal of applied Psychology*, 86 (2001), p. 80. 114

[JDW10] Job, V., Dweck, C. S., and Walton, G. M., Ego depletion—Is it all in your head? Implicit theories about willpower affect self-regulation, *Psychological Science*, 21 (2010), 48

[Jen01] Jenkins, T., Teaching Programming: A Journey from Teacher to Motivator, *Proceedings of the 2nd HEA Conference for the ICS Learning and Teaching Support Network (London, UK)*, July 2001, pp. 53–58. 11, 12, 15, 47, 86

[Jen02] Jenkins, T., On the difficulty of learning to program, *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, 2002, pp. 1–8. 11, 13, 49

[JLDK98] Judge, T. A., Locke, E. A., Durham, C. C., and Kluger, A. N., Dispositional effects on job and life satisfaction: the role of core evaluations, *Journal of applied psychology*, 83 (1998), p. 17. 114

[JLTS11] Jayal, A., Lauria, S., Tucker, A., and Swift, S., Python for Teaching Introductory Programming: A Quantitative Evaluation, *ITALICS*, 10 (2011), pp. 86–90. 93

[JVVDP04] Judge, T. A., Van Vianen, A. E., and De Pater, I. E., Emotional stability, core self-evaluations, and job outcomes: A review of the evidence and an agenda for future research, *Human performance*, 17 (2004), pp. 325–346. 107

[Kaf01] Kafai, Y. B., The educational potential of electronic games: From games-to-teach to games-to-learn, Available from: https://culturalpolicy.uchicago.edu/sites/culturalpolicy.uchicago.edu/files/kafai.pdf, 2001. 108

[Kay10] Kay, J., Robots in the Classroom...and the Dorm Room, *Journal of Computing Sciences in Colleges*, 25 (2010), pp. 128–133. 86

[KB12] Kinnunen, P. and Beth, S., My Program is OK – Am I? Computing Freshman's Experience of Doing Programming Assignments, *Computer Science Education*, 22 (2012), pp. 1 – 28. 12, 13, 67

[KBM73] Krathwohl, D. R., Bloom, B. S., and Masia, B. B., Taxonomy of Educational Objectives, the Classification of Educational Goals Handbook II: Affective Domain, New York, NY: David McKay Co, 1973. 67

[Ke09] Ke, F., A qualitative meta-analysis of computer games as learning tools, *Handbook of research on effective electronic gaming in education*, 1 (2009), pp. 1–32. 108

[Kin01] Kinman, G., Pressure points: A review of research on stressors and strains in UK academics, *Educational psychology*, 21 (2001), pp. 473–492. 22

[Kli05] Kline, R. B., Principles and Practice of Structural Equation Modeling, 2nd ed., New York, NY: The Guilford Press, 2005. 75

[KLMss] Koulouri, T., Lauria, S., and Macredie, R., Teaching Introductory Programming: A Quantitative Evaluation of Different Approaches, *ACM Transactions on Computing Education*, x (in press), 93

[KM06] Kinnunen, P. and Malmi, L., Why Students Drop Out CS1 Courses?, *Proceedings of the 2006 International Computing Education Research Workshop*, 2006, pp. 97 – 108. 13, 16

[KP07] Kelleher, C. and Pausch, R., Using Storytelling to Motivate Programming, *Commun. ACM*, 50 (2007), pp. 58–64, DOI: 10.1145/1272516.1272540. 116

[Kri04] Krippendorff, K., Reliability in content analysis, *Human Communication Research*, 30 (2004), pp. 411–433. 127

[KS10a] Kinnunen, P. and Simon, B., Experiencing Programming Assignments in CS1: The Emotional Toll, *Proceedings of the 6th International Workshop on Computing Education Research*, 2010, pp. 77 – 86. 67

[KS10b] Kinnunen, P. and Simon, B., Experiencing programming assignments in CS1: the emotional toll, *Proceedings of the Sixth international workshop on Computing education research*, 2010, pp. 77–86. 2, 12, 47

[KS12] Kinnunen, P. and Simon, B., My program is ok–am I? Computing freshmen's experiences of doing programming assignments, *Computer Science Education*, 22 (2012), pp. 1–28. 12, 47, 48

[Kum04] Kumar, A. N., Three Years of Using Robots in an Artificial Intelligence Course: Lessons Learned, *J. Educ. Resour. Comput.*, 4 (2004), pp. 2–es, DOI: 10.1145/1083310.1083311. 88

[Laz04] Lazzaro, N., Why we play games: Four keys to more emotion without story, Available from: http://www.xeodesign.com/xeodesign_whyweplaygames.pdf, 2004. 112, 115

[LB12] Lyons, I. M. and Beilock, S. L., When math hurts: math anxiety predicts pain network activation in anticipation of doing math, *PloS one*, 7 (2012), p. e48076. 2, 12

[Lee13] Lee, M. J., How can a social debugging game effectively teach computer programming concepts?, *Proceedings of the ninth annual international ACM conference on International computing education research*, 2013, pp. 181–182. 118

[LG85] Lincoln, Y. S. and Guba, E. G., Naturalistic inquiry, Sage, 1985. 30, 40

[LH05] Ladd, B. and Harcourt, E., Student competitions and bots in an introductory programming course, *Journal of Computing Sciences in Colleges*, 20 (2005), pp. 274–284. 88

[LH11] Livingstone, I. and Hope, A., Next Gen: transforming the UK into the world's leading talent hub for the video games and visual effects industries, Nesta, 2011. 10, 11

[Lik32] Likert, R., A technique for the measurement of attitudes, *Archives of psychology*, 22 (1932), pp. 5–55. 42

[LK77] Landis, J. R. and Koch, G. G., The measurement of observer agreement for categorical data, *Biometrics*, 33 (1977), pp. 159–174. 58

[LK11] Lee, M. J. and Ko, A. J., Personifying programming tool feedback improves novice programmers' learning, *Proceedings of the seventh international workshop on Computing education research*, 2011, pp. 109–116. 112, 115

[LL83] Loftus, G. R. and Loftus, E. F., Mind at play; The psychology of video games, Basic Books, Inc., 1983. 108

[LN10] Lauwers, T. and Nourbakhsh, I., Designing the Finch: Creating a robot aligned to computer science concepts, *AAAI Symposium on Educational Advances in Artificial Intelligence*, 2010. 88

[LNH09] Lauwers, T., Nourbakhsh, I., and Hamner, E., CSbots: design and deployment of a robot designed for the CS1 classroom, *ACM SIGCSE Bulletin*, 41 (2009), pp. 428–432. 87

[LSDB02] Lombard, M., Snyder-Duch, J., and Bracken, C. C., Content analysis in mass communication: Assessment and reporting of intercoder reliability, *Human communication research*, 28 (2002), pp. 587–604. 127

[LW11] Li, F. W. and Watson, C., Game-based concept visualization for learning programming, *Proceedings of the third international ACM workshop on Multimedia technologies for distance learning*, 2011, pp. 37–42. 116, 117

[Mac14] MacLellan, E., How might teachers enable self-confidence? A Review Study, *Educational Review*, 66 (2014), pp. 59 – 74. 82

[MAD+01] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I., and Wilusz, T., A multi-national, multi-institutional study of assessment of programming skills of first-year CS students, *ACM SIGCSE Bulletin*, 33 (2001), pp. 125–180. 2, 11

[Mal80] Malone, T. W., What makes things fun to learn? Heuristics for designing instructional computer games, *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems*, 1980, pp. 162–169. 108

[Mal13] Malmi, L., Doctoral Studies in Computing Education Research: Part 1, *ACM Inroads*, 4 (2013), pp. 18–19, DOI: 10.1145/2537753.2537760. 26, 28

[Mal14] Malmi, L., Doctoral Studies in Computing Education Research—part 2, *ACM Inroads*, 5 (2014), pp. 26–27, DOI: 10.1145/2568195.2568203. 26, 28

[May08] Mayer, R. E., Applying the science of learning: evidence-based principles for the design of multimedia instruction, *American Psychologist*, 63 (2008), pp. 760–769. 2

[MBH+14] Mayer, I., Bekebrede, G., Harteveld, C., Warmelink, H., Zhou, Q., Ruijven, T., Lo, J., Kortmann, R., and Wenzler, I., The research and evaluation of serious games: Toward a comprehensive methodology, *British Journal of Educational Technology*, 45 (2014), pp. 502–527. 117

[MBI+05] McGettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G., and Mander, K., Grand Challenges in Computing: Education - A Summary, *The Computer Journal*, 48 (2005), pp. 42 – 48. 11, 13

[McG81] McGrath, J. E., Dilemmatics: The Study of Research Choices and Dilemmas, *American Behavioral Scientist*, 25 (1981), pp. 179–210. xi, 29, 39

[McG11a] McGonigal, J., Be a gamer, save the world, *Wall Street Journal*, 22 (2011), pp. 01–11. 109

[McG11b] McGonigal, J., Reality is broken: Why games make us better and how they can change the world, Penguin, 2011. 109

[McG12] McGill, M. M., Learning to Program with Personal Robots: Influences on Student Motivation, *ACM Trans. Comput. Educ.*, 12 (2012), pp. 4:1–4:32, DOI: 10.1145/2133797.2133801. 87

[MCGC11] Mampadi, F., Chen, S. Y., Ghinea, G., and Chen, M.-P., Design of adaptive hypermedia learning systems: A cognitive style approach, *Computers & Education*, 56 (2011), pp. 1003–1011. 2

[MD98] Mueller, C. M. and Dweck, C. S., Praise for intelligence can undermine children's motivation and performance, *Journal of personality and social psychology*, 75 (1998), p. 33. 63

[MD04] McKinney, D. and Denton, L. F., Houston, we have a problem: there's a leak in the CS1 affective oxygen tank, *ACM SIGCSE Bulletin*, 36 (2004), pp. 236–239. 12, 47, 145

[MF03] Margolis, J. and Fisher, A., Unlocking the clubhouse: Women in computing, MIT press, 2003. 90

[MK10] Markham, S. A. and King, K. N., Using Personal Robots in CS1: Experiences, Outcomes, and Attitudinal Influences, *Proceedings of the 15th Annual Conference on Innovation and Technology in Computer Science Education (New York, NY, USA)*, 2010, pp. 204–208. 86, 87

[MKB12] Major, L., Kyriacou, T., and Brereton, O., Systematic literature review: teaching novices programming using robots, *IET software*, 6 (2012), pp. 502–513. 87

[ML87] Malone, T. W. and Lepper, M. R., Making learning fun: A taxonomy of intrinsic motivations for learning, *Aptitude, learning, and instruction*, 3 (1987), pp. 223–253. 108

[MLL07] Mcquiggan, S. W., Lee, S., and Lester, J. C., Early prediction of student frustration, *Lecture Notes on Computer Science: Affective Computing and Intelligent Interaction*, 4738 (2007), pp. 698–709. 17

[MM11] Marsh, H. and Martin, A., Academic Self-Concept and Academic Achievement: Relations and Causal Ordering, *British Journal of Educational Psychology*, 81 (2011), pp. 59 – 77. 18, 20, 129

[MR01] Mitra, S. and Rana, V., Children and the Internet: Experiments with minimally invasive education in India, *British Journal of Educational Technology*, 32 (2001), pp. 221–232. 1

[MR+13] McKenney, S., Reeves, T. C., et al., Conducting educational research design, Routledge, 2013. 37

[MS95] March, S. T. and Smith, G. F., Design and Natural Science Research on Information Technology, *Decision Support Systems*, 15 (1995), pp. 251–266, DOI: 10.1016/0167-9236(94)00041-2. 26, 27, 29

[MSS+10] Malmi, L., Sheard, J., Simon, Bednarik, R., Helminen, J., Korhonen, A., Myller, N., Sorva, J., and Taherkhani, A., Characterizing Research in Computing Education: A Preliminary Analysis of the Literature, *Proceedings of the Sixth International Workshop on Computing Education Research (New York, NY, USA)*, 2010, pp. 3–12. 20, 26

[MSS+14] Malmi, L., Sheard, J., Simon, Bednarik, R., Helminen, J., Kinnunen, P., Korhonen, A., Myller, N., Sorva, J., and Taherkhani, A., Theoretical Underpinnings of Computing Education Research: What is the Evidence?, *Proceedings of the Tenth Annual Conference on International Computing Education Research (New York, NY, USA)*, 2014, pp. 27–34. 21, 26

[MSX14] Malliarakis, C., Satratzemi, M., and Xinogalos, S., Educational Games for Teaching Computer Programming, *Research on e-Learning and ICT in Education*, 2014, pp. 87–98. 117

[MT08] Murphy, L. and Thomas, L., Dangers of a fixed mindset: implications of self-theories research for computer science education, *ACM SIGCSE Bulletin*, 40 (2008), pp. 271–275. 47, 48, 51, 63, 67

[Mur00] Murphy, R. R., Using robot competitions to promote intellectual development, *AI magazine*, 21 (2000), p. 77. 88

[NBB67] Nunnally, J. C., Bernstein, I. H., and Berge, J. M. t., Psychometric theory, vol. 226, McGraw-Hill New York, 1967. 71

[NS13] Newton, P. E. and Shaw, S. D., Standards for talking and thinking about validity, *Psychological Methods*, 18 (2013), pp. 301–319. xi, 40, 71, 72

[OB91] Orlikowski, W. J. and Baroudi, J. J., Studying information technology in organizations: Research approaches and assumptions, *Information systems research*, 2 (1991), pp. 1–28. 32, 33

[OHB+14] O'Rourke, E., Haimovitz, K., Ballweber, C., Dweck, C., and Popović, Z., Brain points: a growth mindset incentive structure boosts persistence in an educational game, *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, 2014, pp. 3339–3348. 114

[OMCD06] O'Mara, A. J., Marsh, H. W., Craven, R. G., and Debus, R. L., Do self-concept interventions make a difference? A synergistic blend of construct validation and meta-analysis, *Educational Psychologist*, 41 (2006), pp. 181–206. 18, 21, 107

[Paj92] Pajares, M. F., Teachers beliefs and educational research: Cleaning up a messy construct, *Review of educational research*, 62 (1992), pp. 307–332. 12

[Pap80] Papert, S., Mindstorms: Children, computers, and powerful ideas, Basic Books, Inc., 1980. 125, 131

[Pap08] Papastergiou, M., Are Computer Science and Information Technology Still Masculine Fields? High School Students' Perceptions and Career Choices, *Comput. Educ.*, 51 (2008), pp. 594–608, DOI: 10.1016/j.compedu.2007.06.009. 90

[Pap09] Papastergiou, M., Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation, *Computers & Education*, 52 (2009), pp. 1–12. 109

[Pek06] Pekrun, R., The control-value theory of achievement emotions: Assumptions, corollaries, and implications for educational research and practice, *Educational Psychology Review*, 18 (2006), pp. 315 – 341. xi, 18, 19, 20, 21, 107

[Pet09] Peterson, M., The use of computerized games and simulations in computer-assisted language learning: A meta-analysis of research, *Simulation & Gaming*, 41 (2009), 109, 125

[PJM10] Peyton-Jones, S. and Mitchell, B., The Collapse of Computing Education in Schools, *SIP: The Journal of the Parliamentary and Scientific Committee*, 67 (2010), pp. 39–40. 10

[PL92] Parker, L. E. and Lepper, M. R., Effects of fantasy contexts on children's learning and motivation: Making learning more fun, *Journal of Personality and Social Psychology*, 62 (1992), p. 625. 108

[PP13] Peters, A.-K. and Pears, A., Engagement in Computer Science and IT – What! A Matter of Identity?, *Learning and Teaching in Computing and Engineering Conference*, 2013, pp. 114 – 121. 13

[Pre05] Prensky, M., Computer games and learning: Digital game-based learning, *Handbook of computer game studies*, 18 (2005), pp. 97–122. 109

[PS10] Pekrun, R. and Stephens, E. J., Achievement Emotions: A Control-Value Approach, *Social and Personality Psychology Compass*, 4 (2010), pp. 238 – 255. xi, 18, 20, 21

[PS13] Porter, L. and Simon, B., Retaining nearly one-third more majors with a trio of instructional best practices in CS1, *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, 2013, pp. 165 – 170. 3, 11

[PSAS13] Peck, T. C., Seinfeld, S., Aglioti, S. M., and Slater, M., Putting yourself in the skin of a black avatar reduces implicit racial bias, *Consciousness and cognition*, 22 (2013), pp. 779–787. 111, 113

[RAB12] Richardson, M., Abraham, C., and Bond, R., Psychological correlates of university students' academic performance: a systematic review and meta-analysis, *Psychological bulletin*, 138 (2012), pp. 353–387. 18

[RCV09] Ritterfeld, U., Cody, M., and Vorderer, P., Serious games: Mechanisms and effects, Routledge, 2009. 109

[Rep12] Repenning, A., Programming goes back to school, *Communications of the ACM*, 55 (2012), pp. 38–40. 130

[RFK$^+$09] Resnick, M., Flanagan, M., Kelleher, C., MacLaurin, M., Ohshima, Y., Perlin, K., and Torres, R., Growing up programming: democratizing the creation of dynamic, interactive media, *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, 2009, pp. 3293–3296. 130

[RGD12] Rattan, A., Good, C., and Dweck, C. S., It's okNot everyone can be good at math: Instructors with an entity theory comfort (and demotivate) students, *Journal of Experimental Social Psychology*, 48 (2012), pp. 731–737. 63

[Rie96] Rieber, L. P., Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games, *Educational technology research and development*, 44 (1996), pp. 43–58. 108

[RJBS07] Randolph, J. J., Julnes, G., Bednarik, R., and Sutinen, E., A comparison of the methodological quality of articles in computer science education journals and conference proceedings, *Computer Science Education*, 17 (2007), pp. 263–274. 31

[RJSL08a] Randolph, J., Julnes, G., Sutinen, E., and Lehman, S., A Methodological Review of Computer Science Education Research, *Information Technology Education*, 7 (2008), pp. 135 – 162. 67, 81

[RJSL08b] Randolph, J., Julnes, G., Sutinen, E., and Lehman, S., A methodological review of computer science education research, *Journal of Information Technology Education: Research*, 7 (2008), pp. 135–162. 31

[RM01] Rieber, L. P. and Matzko, M. J., Serious design for serious play in physics, *Educational Technology*, 41 (2001), pp. 14–24. 108

[RMWW92] Randel, J. M., Morris, B. A., Wetzel, C. D., and Whitehill, B. V., The effectiveness of games for educational purposes: A review of recent research, *Simulation & gaming*, 23 (1992), pp. 261–276. 108, 109

[Rob00] Roberts, E., Strategies for encouraging individual achievement in introductory computer science courses, *ACM SIGCSE Bulletin*, 32 (2000), pp. 295–299. 88

[Rob10] Robins, A., Learning edge momentum: A new account of outcomes in CS1, *Computer Science Education*, 20 (2010), pp. 37–71. 15

[Rob12] Robins, A., Learning Edge Momentum, Encyclopedia of the Sciences of Learning, Springer, pp. 1845–1848. 15

[RRZMndM15] Rubio, M. A., Romero-Zaliz, R., Mañoso, C., and de Madrid, A. P., Closing the Gender Gap in an Introductory Programming Course, *Computers & Education*, 82 (2015), pp. 409–420, DOI: 10.1016/j.compedu.2014.12.003. 149

[RS10a] Rogerson, C. and Scott, E., The Fear Factor: How it Affects Students Learning to Program in a Tertiary Environment, *Information Technology Education*, 9 (2010), pp. 147 – 171. 12, 17, 67

[RS10b] Rogerson, C. and Scott, E., The fear factor: How it affects students learning to program in a tertiary environment, *Journal of Information Technology Education: Research*, 9 (2010), pp. 147–171. 2, 12, 47

[SBE83] Soloway, E., Bonar, J., and Ehrlich, K., Cognitive strategies and looping constructs: An empirical study, *Communications of the ACM*, 26 (1983), pp. 853–860. 2, 11

[SBG04] Straub, D., Boudreau, M.-C., and Gefen, D., Validation Guidelines for IS Positivist Research, pp. 380 – 427. xi, 71, 72, 80, 81

[SCC02] Shadish, W. R., Cook, T. D., and Campbell, D. T., Experimental and quasi-experimental designs for generalized causal inference, Wadsworth Cengage learning, 2002. 40

[Sch10] Schelle, J., Design Outside of the Box: The Future of Games, Presentation at the DICE Summit. Available from: http://www.dicesummit.org/video_gallery/video_gallery_2010.asp, 2010. 109

[SCL+ss] Scott, M. J., Counsell, S., Lauria, S., Swift, S., Tucker, A., Shepperd, M., and Ghinea, G., Enhancing Practice and Achievement in Introductory Programming with a Robot Olympics, *IEEE Transactions on Education* (Accepted, In Press), pp. 1–6, DOI: 10.1109/TE.2013.2288700.

[Sco13] Scott, M. J., Projective Identity and Procedural Rhetoric in Educational Multimedia: Towards the Enrichment of Programming Self-concept and Growth Mindset with narrative reinforcement, *Proceedings of the 21st ACM International Conference on Multimedia (Barcelona, Spain)*, October 2013, pp. 1031–1034.

[SDR+15] Spacco, J., Denny, P., Richards, B., Babcock, D., Hovemeyer, D., Moscola, J., and Duvall, R., Analyzing Student Work Patterns Using Programming Exercise Data, *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (New York, NY, USA)*, 2015, pp. 18–23. 148

[SFS+06] Simon, Fincher, S, Robins, A, Baker, B, Box, I, Cutts, Q, de Raadt, M, Haden, P, Hamer, J, Hamilton, M, Lister, R, Petre, M, Sutton, K, Tolhurst, D, Tutty, and J, Predictors of success in a first programming course, *Proceedings of the 8th Australasian Conference on Computing Education*, 2006, pp. 189 – 196. 11

[SFSR13] Spacco, J., Fossati, D., Stamper, J., and Rivers, K., Towards Improving Programming Habits to Create Better Computer Science Course Outcomes, *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education (New York, NY, USA)*, 2013, pp. 243–248. 148

[SG13a] Scott, M. J. and Ghinea, G., Educating Programmers: A Reflection on Barriers to Deliberate Practice, *Proceedings of the 2nd HEA STEM Conference (Birmingham, UK)*, April 2013, pp. 28–33.

[SG13b] Scott, M. J. and Ghinea, G., Implicit Theories of Programming Aptitude as a Barrier to Learning to Code: Are They Distinct from Intelligence?, *Proceedings of the 18th ACM Annual Conference on Innovation and Technology in Computer Science Education (Kent, UK)*, July 2013, p. 347.

[SG13c] Scott, M. J. and Ghinea, G., Integrating narrative reinforcement into the Programming Lab: Exploring the 'Projective Identity' Hypothesis, *Proceedings of 43rd ACM Technical Symposium on Computer Science Education (Denver, CO, USA)*, March 2013, pp. 119–122.

[SG14a] Scott, M. J. and Ghinea, G., On the Domain-Specificity of Mindsets: The Relationship Between Aptitude Beliefs and Programming Practice, *IEEE Transactions on Education*, 57 (2014), pp. 169–174, DOI: 10.1109/TE.2013.2288700.

[SG14b] Scott, M. J. and Ghinea, G., Measuring Enrichment: The Assembly and Validation of an Instrument to Assess Student Self-beliefs in CS1, *Proceedings of the 10th Annual ACM Conference on International Computing Education Research (Glasgow, Scotland)*, August 2014, pp. 123–130.

[SGew] Scott, M. J. and Ghinea, G., Enriching the Self-Concept and Mindset of Novice Programmers using Game-based Fantasy Role-Play: A Review of Existing Games, *British Journal of Educational Technology* (Under Review), pp. 1–6, DOI: 10.1109/TE.2013.2288700.

[Sha06] Shaffer, D. W., Epistemic frames for epistemic games, *Computers & education*, 46 (2006), pp. 223–234. 108

[SHCD09] Stump, G., Husman, J., Chung, W.-T., and Done, A., Student beliefs about intelligence: Relationship to learning, *Frontiers in Education Conference, 2009. FIE'09. 39th IEEE*, 2009, pp. 1–6. 48

[She11a] Sheldon, L., The multiplayer classroom: Designing coursework as a game, Cengage Learning, 2011. 109, 111

[She11b] Shepperd, M., Group project work from the outset: An in-depth teaching experience report, *Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on*, 2011, pp. 361–370. 92

[SHM+08a] Simon, B., Hanks, B., Murphy, L., Fitzgerald, S., McCauley, R., Thomas, L., and Zander, C., Saying Isn't Necessarily Believing: Influencing Self-Theories in Computing, *Proceedings of the 4th Int. Workshop on Computing Education Research*, 2008, pp. 173 – 184. 67

[SHM+08b] Simon, B., Hanks, B., Murphy, L., Fitzgerald, S., McCauley, R., Thomas, L., and Zander, C., Saying isn't necessarily believing: influencing self-theories in computing, *Proceedings of the Fourth international Workshop on Computing Education Research*, 2008, pp. 173–184. 49

[Sim96] Simon, H. A., The sciences of the artificial, MIT press, 1996. 26

[Sit11] Sitzmann, T., A meta-analytic examination of the instructional effectiveness of computer-based simulation games, *Personnel Psychology*, 64 (2011), pp. 489–528. 109, 125

[SK07a] Simons, K. D. and Klein, J. D., The impact of scaffolding and student achievement levels in a problem-based learning environment, *Instructional Science*, 35 (2007), pp. 41–72. 15

[SK07b] Simons, K. D. and Klein, J. D., The impact of scaffolding and student achievement levels in a problem-based learning environment, *Instructional Science*, 35 (2007), pp. 41–72. 91, 145

[Squ03] Squire, K., Video games in education, *Int. J. Intell. Games & Simulation*, 2 (2003), pp. 49–62. 108

[Squ11] Squire, K., Video games and learning, Teaching and participatory culture in the digital age. New York, NY: Teachers College Print. Cerca con Google, 2011. 108

[SS80] Swann, W. B. and Snyder, M., On translating beliefs into action: Theories of ability and their application in an instructional setting., *Journal of Personality and Social Psychology*, 38 (1980), p. 879. 12, 16

[SSD99] Springer, L., Stanne, M. E., and Donovan, S. S., Effects of small-group learning on undergraduates in science, mathematics, engineering, and technology: A meta-analysis, *Review of educational research*, 69 (1999), pp. 21–51. 91

[SSHL09] Sheard, J., Simon, S., Hamilton, M., and Lönnberg, J., Analysis of Research into the Teaching and Learning of Programming, *Proceedings of the Fifth International Workshop on Computing Education Research Workshop (New York, NY, USA)*, 2009, pp. 93–104. 20

[Sui14] Suits, B., The grasshopper: Games, life and utopia, Broadview Press, 2014. 108

[SVKT08] Shabalina, O., Vorobkalov, P., Kataev, A., and Tarasenko, A., Educational games for learning programming languages, *Information Science and Computing*, 2008, pp. 79–83. 109

[Swi08] Swink, S., Game Feel: A Game Designer's Guide to Virtual Sensation, Morgan Kaufmann, 2008. 112, 115

[TBC12] Thota, N., Berglund, A., and Clear, T., Illustration of paradigm pluralism in computing education research, *Proceedings of the Fourteenth Australasian Computing Education Conference-Volume 123*, 2012, pp. 103–112. 26

[TD98] Tooley, J. and Darby, D., Educational research: a critique, Office for Standards in Education, 1998. 30

[TD13] Tew, A. and Dorn, B., The Case for Validated Tools in Computer Science Education Research, *Computer*, 46 (2013), pp. 60–66. 63, 66, 71, 81, 145

[TFDW11] Tobias, S., Fletcher, J., Dai, D. Y., and Wind, A. P., Review of research on computer games, *Computer games and instruction*, 2011, pp. 127–222. 108, 109, 125

[TG11] Tew, A. E. and Guzdial, M., The FCS1: A Language Independent Assessment of CS1 Knowledge, *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, 2011, pp. 111 – 116. 2, 11, 67

[THB08] Tychsen, A., Hitchens, M., and Brolund, T., Character play: the use of game characters in multi-player role-playing games across platforms, *Computers in Entertainment (CIE)*, 6 (2008), pp. 22:1–22:24. 112, 114

[Tig10] Tight, M., Are academic workloads increasing? The post-war survey evidence in the UK, *Higher Education Quarterly*, 64 (2010), pp. 200–215. 22, 107

[VA04] Verner, I. M. and Ahlgren, D. J., Robot contest as a laboratory for experiential engineering education, *ACM Journal on Educational Resources in Computing*, 4 (2004), p. 2. 88

[Val04] Valentine, D. W., CS Educational Research: A Meta-Analysis of SIGCSE Technical Symposium Proceedings, *Proceedings of the 35th ACM Technical Symposium on Computer Science Education*, 2004, pp. 255 – 259. 20, 67

[VDC04] Valentine, J. C., DuBois, D. L., and Cooper, H., The relation between self-beliefs and academic achievement: A meta-analytic review, *Educational Psychologist*, 39 (2004), pp. 111–133. 107

[Ver13] Verner, I., Characteristics of Student Engagement in Robotics, Intelligent Robotics Systems: Inspiring the NEXT (K. Omar, M. Nordin, P. Vadakkepat, A. Prabuwono, S. Abdullah, J. Baltes, S. Amin, W. Hassan, and M. Nasrudin, eds.), Springer Berlin Heidelberg, 92

[VJ05] Ventura Jr, P. R., Identifying predictors of success for an objects-first CS1, *Computer Science Education*, 15 (2005), pp. 223–243. 11

[VMM14] Vahldick, A., Mendes, A. J., and Marcelino, M. J., A review of games designed to improve introductory computer programming competencies, *IEEE Frontiers in Education Conference*, Oct 2014, pp. 1–7. 117

[VVCB+06] Vogel, J. J., Vogel, D. S., Cannon-Bowers, J., Bowers, C. A., Muse, K., and Wright, M., Computer gaming and interactive simulations for learning: A meta-analysis, *Journal of Educational Computing Research*, 34 (2006), pp. 229–243. 108

[Vyg80] Vygotsky, L. S., Mind in society: The development of higher psychological processes, Harvard university press, 1980. 15

[Wes10] Westland, J. C., Lower bounds on sample size in structural equation modeling, *Electronic Commerce Research and Applications*, 9 (2010), pp. 476–487. 73

[WEY+97] Wigfield, A., Eccles, J. S., Yoon, K. S., Harold, R. D., Arbreton, A. J. A., and Freedman-Doan, C., Change in children's competence beliefs and subjective task values across the elementary school years: A 3-year study, pp. 451 – 469. 70

[Wie05] Wiedenbeck, S., Factors affecting the success of non-majors in learning to program, *Proceedings of the 1st Int. Workshop Computing Education Research*, 2005, pp. 13 – 24. 19

[Win96] Winslow, L. E., Programming Pedagogy–A Psychological Overview, *SIGCSE Bulletin*, 28 (1996), pp. 17–22, DOI: 10.1145/234867.234872. 62

[WML88] Wigfield, A., Meece, and L, J., Math anxiety in elementary and secondary school students, *Journal of Educational Psychology*, 80 (1988), pp. 210 – 216. 70

[WS01] Wilson, B. C. and Shrock, S., Contributing to success in an introductory computer science course: A study of twelve factors, *SIGCSE Bulletin*, 33 (2001), pp. 184 – 188. 11, 19

[WVNVOVDS13] Wouters, P., Van Nimwegen, C., Van Oostendorp, H., and Van Der Spek, E. D., A meta-analysis of the cognitive and motivational effects of serious games, *Journal of Educational Psychology*, 105 (2013), p. 249. 109, 125

[YB07] Yee, N. and Bailenson, J., The Proteus effect: The effect of transformed self-representation on behavior, *Human communication research*, 33 (2007), pp. 271–290. 111, 113, 116, 131

[YD12] Yeager, D. S. and Dweck, C. S., Mindsets that promote resilience: When students believe that personal characteristics can be developed, *Educational Psychologist*, 47 (2012), pp. 302–314. 48, 51, 62

[YSC+12] Young, M. F., Slota, S., Cutter, A. B., Jalette, G., Mullin, G., Lai, B., Simeoni, Z., Tran, M., and Yukhymenko, M., Our princess is in another castle a review of trends in serious gaming for education, *Review of Educational Research*, 82 (2012), pp. 61–89. 108

[Zim02] Zimmerman, B. J., Becoming a self-regulated learner: An overview, *Theory into practice*, 41 (2002), pp. 64–70. 2

[ZL10] Zichermann, G. and Linder, J., Game-based marketing: inspire customer loyalty through rewards, challenges, and contests, John Wiley & Sons, 2010. 109, 111