Computing Education and Learning Technology
Research Group

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

Dr Paul Denny
Associate Professor
School of Computer Science
p.denny@auckland.ac.nz

Overview

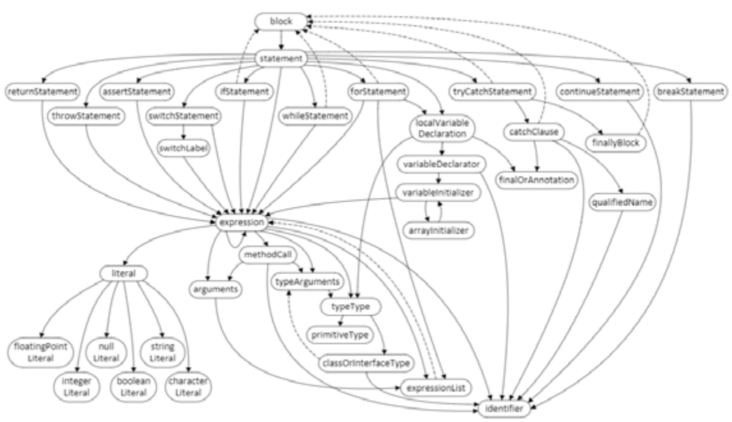
- What is Computing Education & Learning Technology research?
- Why is it an interesting area of research?
- A few examples
  - student projects (including COMPSCI 747)
  - leveraging existing expertise in Computer Science
- Overarching research questions
- Two examples
  - Specific research questions

What is it?

- Distinct from “teaching”
  - Teaching is helping others acquire knowledge and develop skills in a discipline
  - Research is creating new knowledge and exploring new ideas

What is it?

- Distinct from “teaching”
  - Teaching is helping others acquire knowledge and develop skills in a discipline
  - Research is creating new knowledge and exploring new ideas
- Computing Education Research:
  - the study of how people learn and teach computing
  - the goal is to help students learn, and teachers teach, more effectively




```
int sum = 0;

if(a==2*(a/2)) {
    if(b!=2*(b/2)) {
        sum = a+b;
    }
} else {
    if(b==2*(b/2)) {
        sum = a+b;
    }
}

return sum;
```

What is it?

- Distinct from “teaching”
  - Teaching is helping others acquire knowledge and develop skills in a discipline
  - Research is creating new knowledge and exploring new ideas
- Computing Education Research:
  - the study of how people learn and teach computing
  - the goal is to help students learn, and teachers teach, more effectively
- Learning Technology Research:
  - designing and evaluating tools for learning
  - covers the broader use of technology in teaching, learning and education across disciplines

Why research education?

- To have a positive impact in the world
  - Better outcomes for learners

Why research education?

- To have a positive impact in the world
  - Better outcomes for learners
- Practical application of technology
  - A clear need and an enormous user base

Why research education?

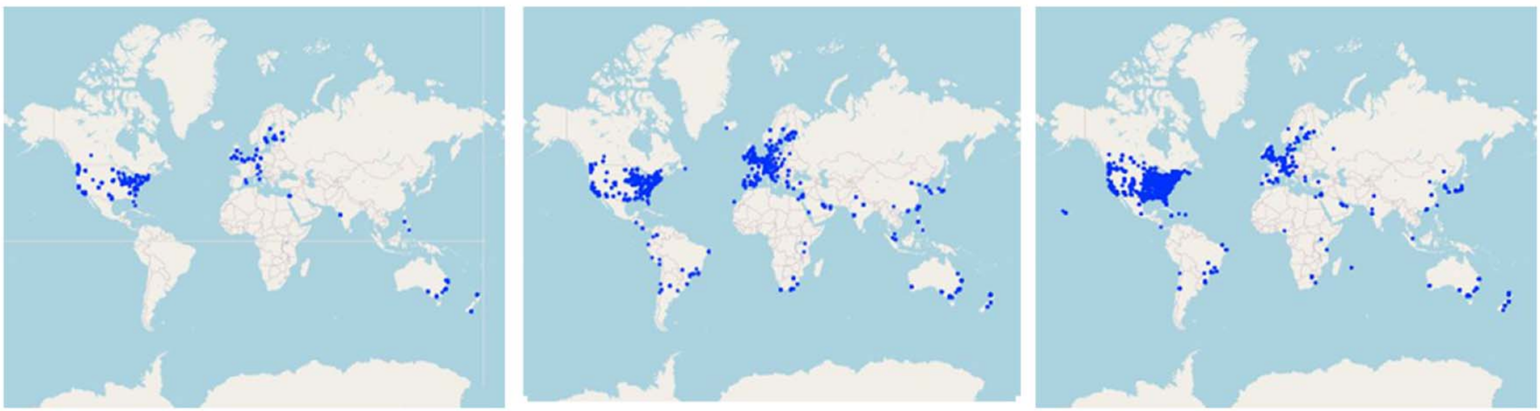
- To have a positive impact in the world
  - Better outcomes for learners
- Practical application of technology
  - A clear need and an enormous user base
- Big business
  - "Computational Thinking" is an essential 21st century skill, yet there are few people who can help others to develop those skills.
    - Constant need to train staff in technology
    - Increasing need for non-CS people to program
    - Increasing integration of programming skills into school curriculum
    - Increasing number of companies involved in technology to support education, and education about CS.

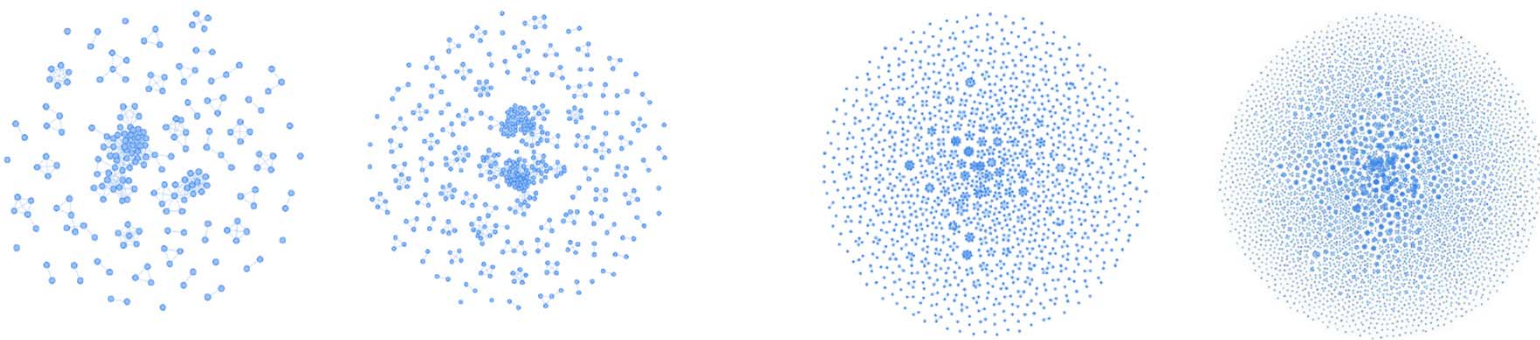
A few student projects

- Researchers apply their expertise from many areas of CS

A few student projects

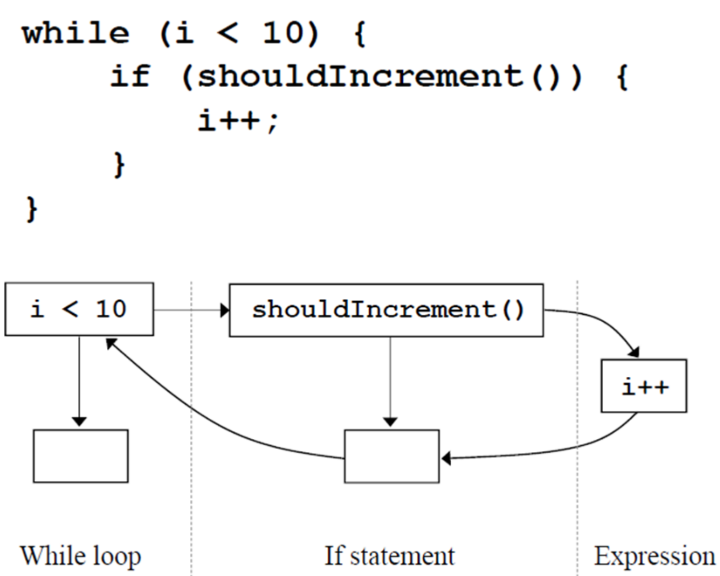
- Researchers apply their expertise from many areas of CS
  - **graph theory** (e.g. bibliometric analysis - James)

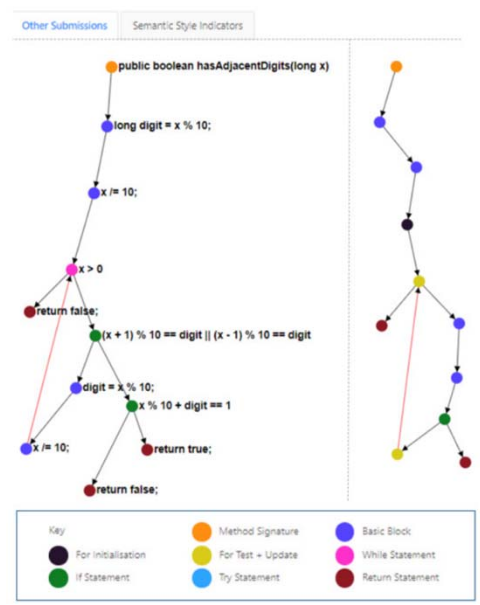


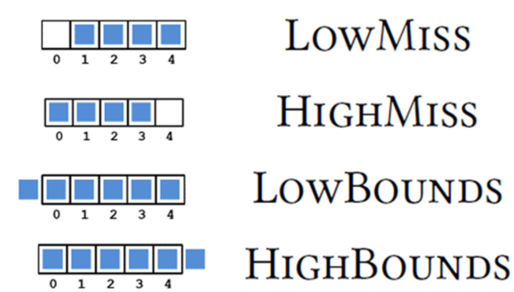


A few student projects

- Researchers apply their expertise from many areas of CS
  - graph theory (e.g. bibliometric analysis - James)
  - **program analysis**
    - static analysis (e.g. generating ASTs for visualising control flow - Lucy and Robert)
    - dynamic analysis (e.g. classification of array access errors - Liam)

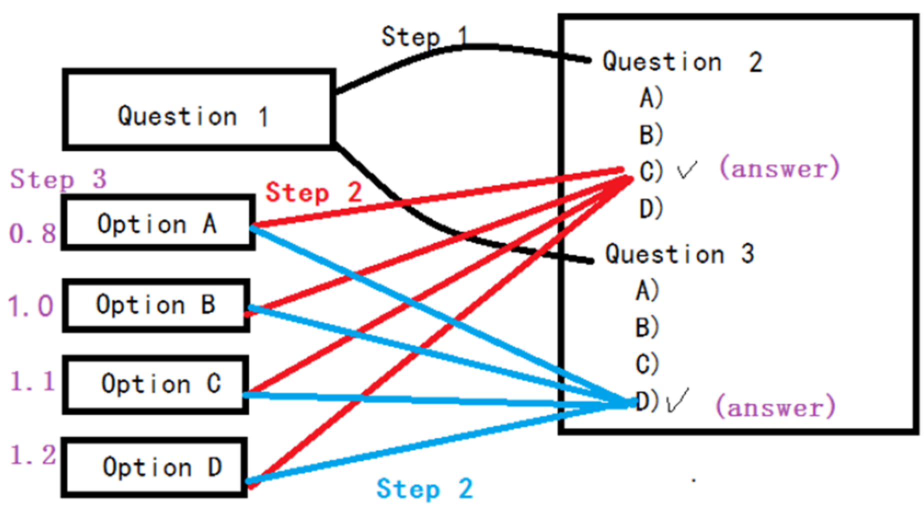


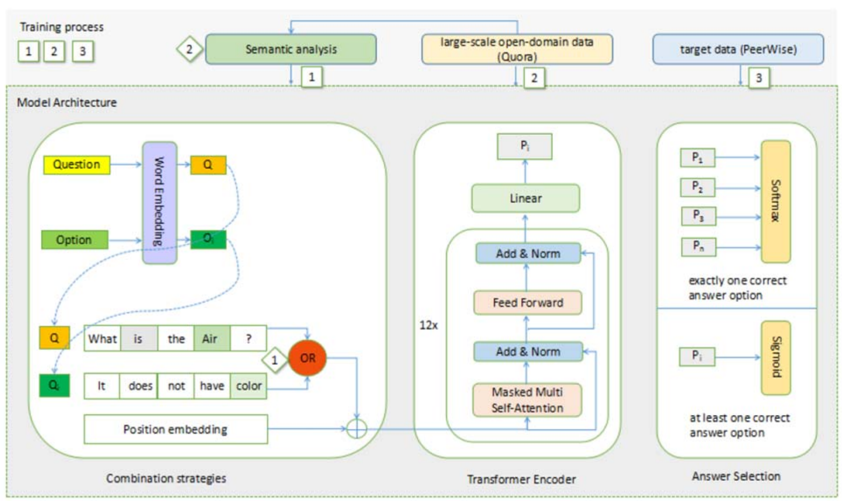




A few student projects

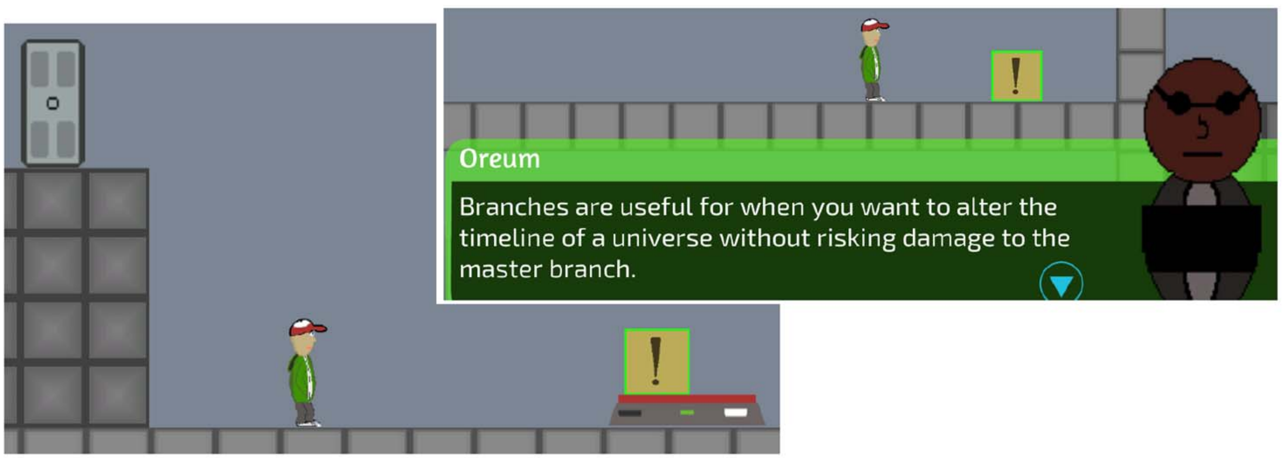
- Researchers apply their expertise from many areas of CS
  - graph theory (e.g. bibliometric analysis - James)
  - program analysis
    - static analysis (e.g. generating ASTs for visualising control flow - Lucy and Robert)
    - dynamic analysis (e.g. classification of array access errors - Liam)
  - **machine learning/AI** (e.g. NLP for question answering - Bill)

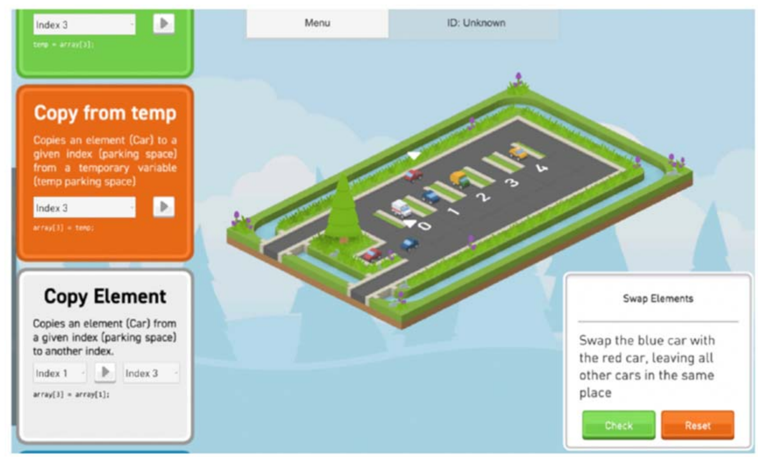




A few student projects

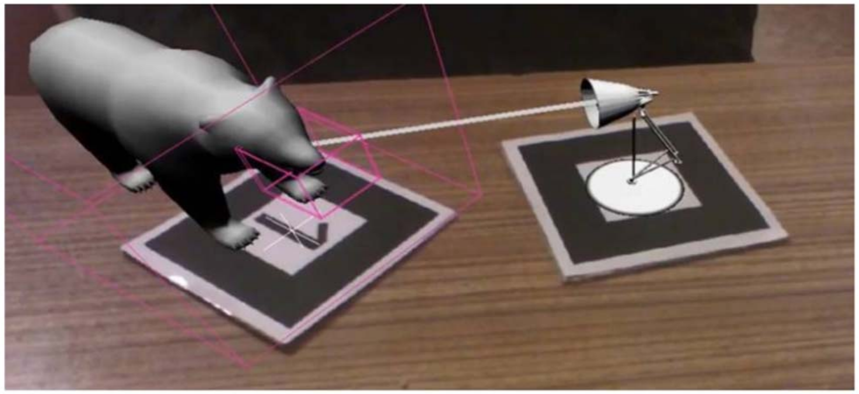
- Researchers apply their expertise from many areas of CS
  - graph theory (e.g. bibliometric analysis - James)
  - program analysis
    - static analysis (e.g. generating ASTs for visualising control flow - Lucy and Robert)
    - dynamic analysis (e.g. classification of array access errors - Liam)
  - machine learning/AI (e.g. NLP for question answering - Bill)
  - **game design** (e.g. teaching version control, data structures – Kevin, Simon)





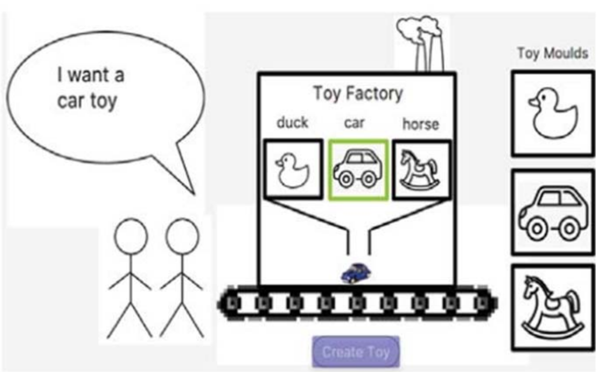
A few student projects

- Researchers apply their expertise from many areas of CS
  - graph theory (e.g. bibliometric analysis - James)
  - program analysis
    - static analysis (e.g. generating ASTs for visualising control flow - Lucy and Robert)
    - dynamic analysis (e.g. classification of array access errors - Liam)
  - machine learning/AI (e.g. NLP for question answering - Bill)
  - game design (e.g. teaching version control, data structures – Kevin, Simon)
  - **VR/AR** (e.g. visualising object transformations for computer graphics - Thomas)



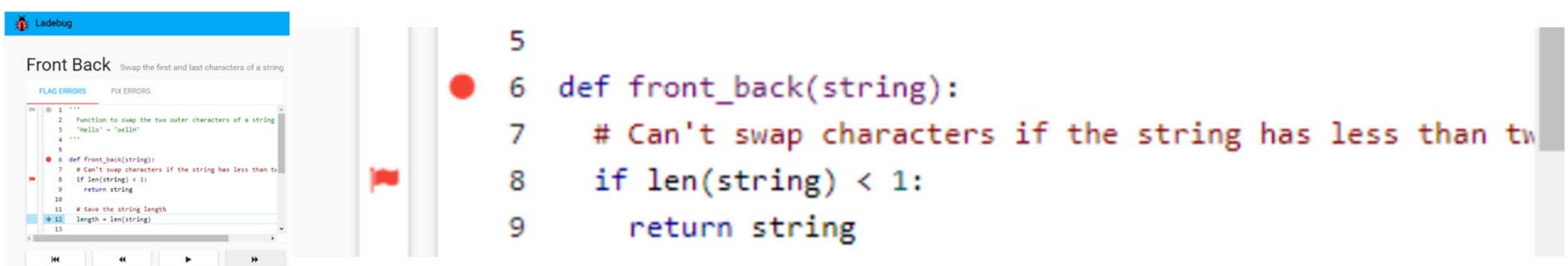
A few student projects

- Researchers apply their expertise from many areas of CS
  - graph theory (e.g. bibliometric analysis - James)
  - program analysis
    - static analysis (e.g. generating ASTs for visualising control flow - Lucy and Robert)
    - dynamic analysis (e.g. classification of array access errors - Liam)
  - machine learning/AI (e.g. NLP for question answering - Bill)
  - game design (e.g. teaching version control, data structures – Kevin, Simon)
  - VR/AR (e.g. visualising object transformations for computer graphics - Thomas)
  - **software design** (e.g. using metaphors for teaching design patterns - Zain)



A few student projects

- Researchers apply their expertise from many areas of CS
  - graph theory (e.g. bibliometric analysis - James)
  - program analysis
    - static analysis (e.g. generating ASTs for visualising control flow - Lucy and Robert)
    - dynamic analysis (e.g. classification of array access errors - Liam)
  - machine learning/AI (e.g. NLP for question answering - Bill)
  - game design (e.g. teaching version control, data structures – Kevin, Simon)
  - VR/AR (e.g. visualising object transformations for computer graphics - Thomas)
  - software design (e.g. using metaphors for teaching design patterns - Zain)
  - **programming** (e.g. interactive tool for teaching debugging skills - Emma and Liz)



A few student projects

- Researchers apply their expertise from many areas of CS
  - graph theory (e.g. bibliometric analysis - James)
  - program analysis
    - static analysis (e.g. generating ASTs for visualising control flow - Lucy and Robert)
    - dynamic analysis (e.g. classification of array access errors - Liam)
  - machine learning/AI (e.g. NLP for question answering - Bill)
  - game design (e.g. teaching version control, data structures – Kevin, Simon)
  - VR/AR (e.g. visualising object transformations for computer graphics - Thomas)
  - software design (e.g. using metaphors for teaching design patterns - Zain)
  - programming (e.g. interactive tool for teaching debugging skills - Emma and Liz)
  - **computer systems** (e.g. compiler error messages - Dave)

```
if (a < 0) || (a > 100)
    error = true;
```

```
if (a < 0) || (a > 100)
           ^^
Syntax error on token "||", if expected

1 error
```

COMPSCI 747

- Graduate course in Computing Education

Example

- Liam Rigby's paper
  - COMPSCI747 project
  - Australasian Computing Education Conference (ACE) 2020
  - Dynamic analysis of large existing code dataset

A Miss is as Good as a Mile: Off-By-One Errors and Arrays in an Introductory Programming Course

Liam Rigby
The University of Auckland
New Zealand
lrig521@aucklanduni.ac.nz

Paul Denny
The University of Auckland
New Zealand
paul@cs.auckland.ac.nz

Andrew Luxton-Reilly
The University of Auckland
New Zealand
a.luxton-reilly@auckland.ac.nz

ABSTRACT

Loops and arrays are fundamental CS1 concepts, but ones that can be problematic for novice programmers. In this research, we investigate *off-by-one* errors – logic errors where loops perform one too few or one too many iterations – in code using an indexed loop over an array. We classify off-by-one errors, and explore the prevalence of each type, by analyzing a large set of code submissions from students in a first year programming course as they tackle a sequence of exercises. We describe an approach to reliably identify off-by-one errors through dynamic analysis, and find that off-by-one errors are both common and persist across exercises. We also show that students infrequently choose to iterate over an array in reverse, but when they do they more commonly encounter off-by-one errors. We conclude that teaching material should explicitly focus student attention on boundary cases, and should provide more examples that iterate through arrays in reverse.

LowMiss
(first element, at index *0*, is missed)

```
for (i = 1; i < length; i++) {
    sum += array[i];
}
```

HighMiss
(last element, at index *length-1*, is missed)

```
for (i = 0; i < length - 1; i++) {
    sum += array[i];
}
```

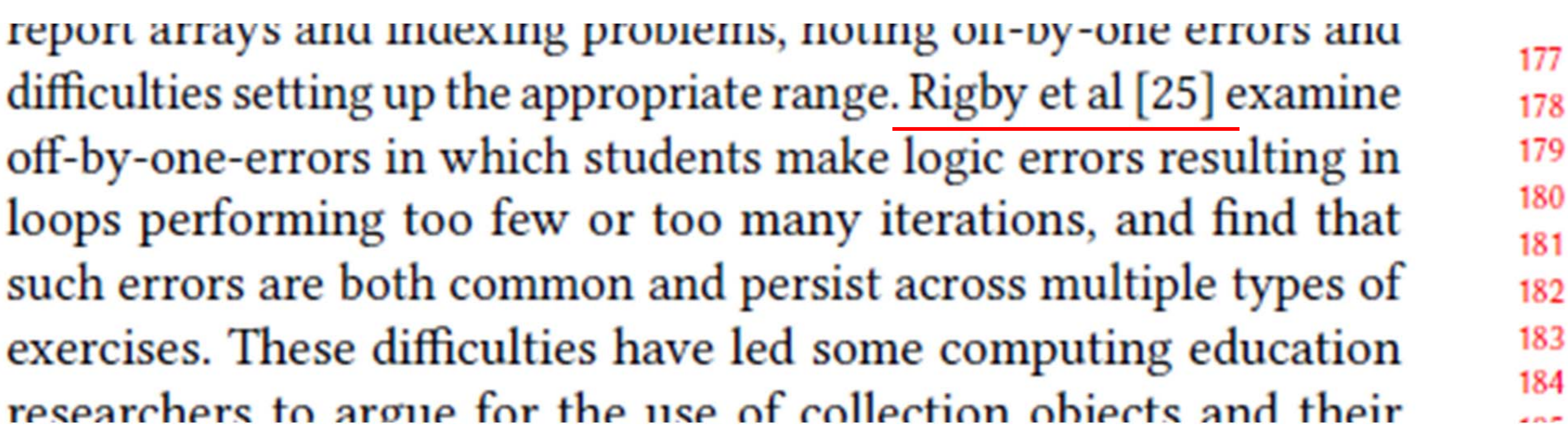
LowBounds
(invalid element, at index *-1*, is accessed)

```
for (i = -1; i < length; i++) {
    sum += array[i];
```

Example

- Liam Rigby's paper
  - COMPSCI747 project
  - Australasian Computing Education Conference (ACE) 2020
  - Dynamic analysis of large existing code dataset

was reviewing recently...



report arrays and indexing problems, noting off-by-one errors and difficulties setting up the appropriate range. Rigby et al [25] examine off-by-one-errors in which students make logic errors resulting in loops performing too few or too many iterations, and find that such errors are both common and persist across multiple types of exercises. These difficulties have led some computing education researchers to argue for the use of collection objects and their

Example

- Liam's paper
  - CS747 project
  - Published in ACE 2
  - Analysed an existin

was reviewing th

report arrays and
difficulties setting
off-by-one-errors
loops performin
such errors are b
exercises. These
researchers to a

Liam Rigby

Paul Denny

Re: First citation (maybe)

You replied to this message on 28/04/2020 12:16 PM.

An exciting time

- Rapid growth in adoption of learning tools
- Vast amounts of data collected on how people learn

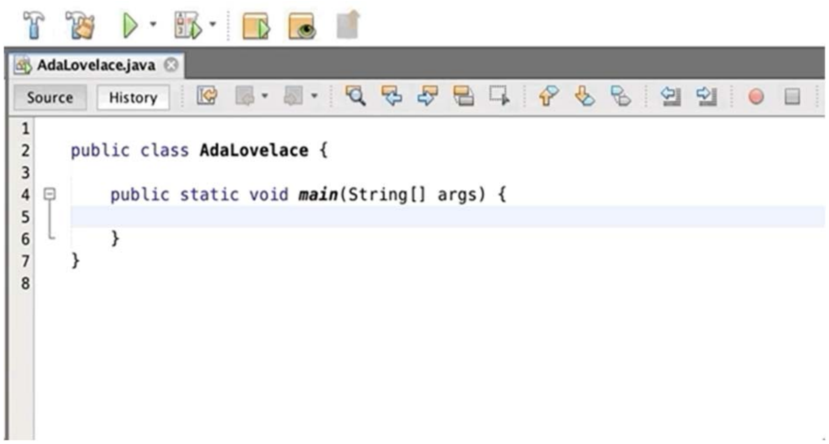
Anant Agarwal (MIT / EdX)

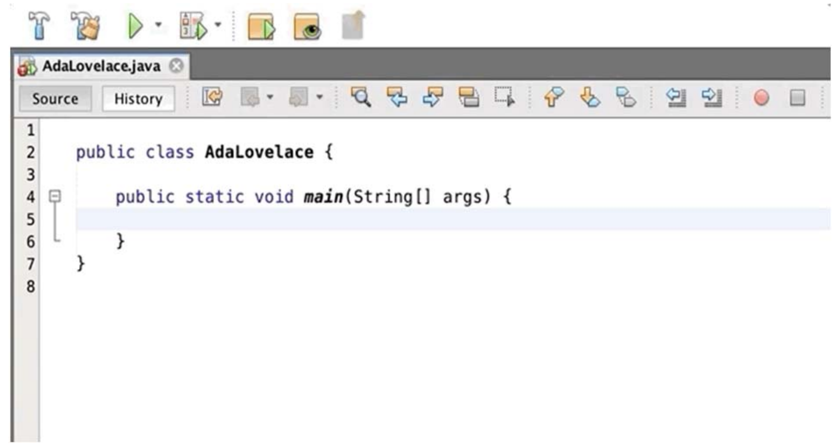
"Data collection for educational research is one of the key goals of EdX.... we gather huge amounts of data.... all this rich data will be available to researchers.... to understand how people really learn and we can help synthesize a better educational experience"

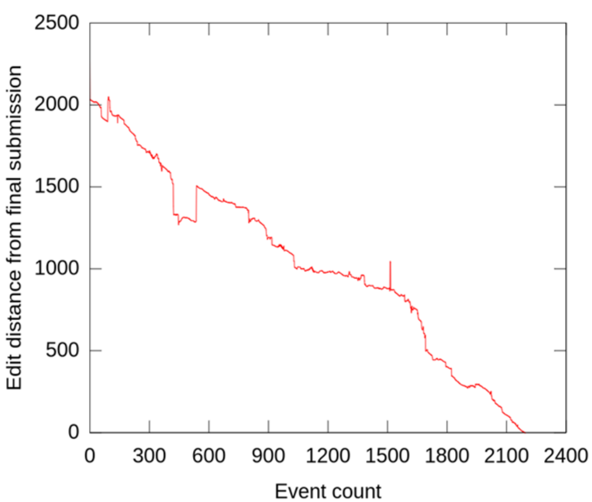
Daphne Koller (Stanford / Coursera)

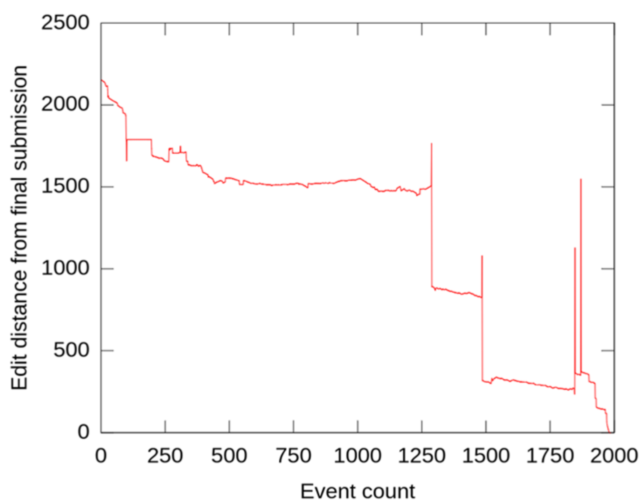
"Tremendous opportunities.... every click, every homework submission, every forum post, from tens of thousands of students.... turn the study of human learning from the hypothesis driven mode to the data driven mode"

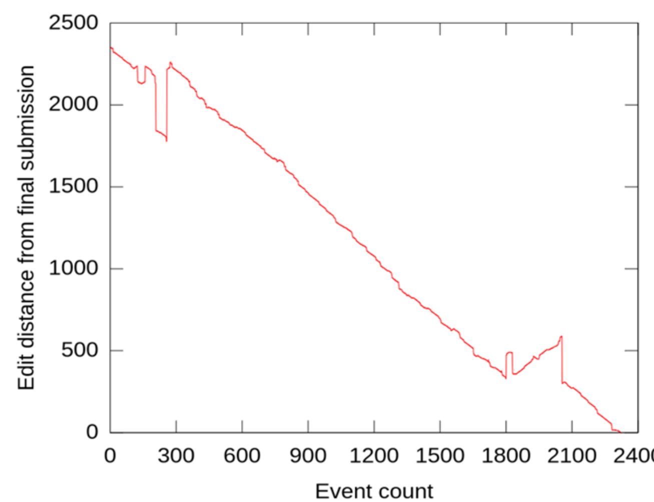
Example: keystroke vs submission data











Courtesy: Juho Leinonen (University of Helsinki)

Computing Education Research

- Overarching research questions:
  - How do people learn computing?
  - How do teachers teach and assess computing?
  - How can people learn computing more effectively?
  - How can teachers teach computing more effectively?
  - How can access to computing education be improved?
  - How can computing education be delivered equitably to all?
  - How can learning technologies teach computing?
  - How does computing education affect people's lives?
  - What are the societal costs of computing illiteracy?
  - What does it mean to know computing?
  - What can be taught about computing to learners of different ages?

Two examples

- Two recent studies
- Feedback and learner behavior
  - Example 1
    - Computing education: *Compiler error messages*
  - Example 2
    - Learning technology: *Influencing (positive) behaviours*

Example 1: Compiler error messages

Error Message Readability and Novice Debugging Performance

Paul Denny
University of Auckland
Auckland, New Zealand
paul@cs.auckland.ac.nz

James Prather
Abilene Christian University
Abilene, Texas, USA
james.prather@acu.edu

Brett A. Becker
University College Dublin
Dublin, Ireland
brett.becker@ucd.ie

ABSTRACT

It is well known that programming error messages can be notoriously difficult for novices to understand, hampering progress and leading to frustration. In response, researchers have explored various approaches for enhancing such messages, yet results from this active strand of research are currently mixed. Direct comparisons of results between studies is challenging as these typically investigate different kinds of message enhancements and report results using different metrics. In addition, many prior studies have involved code writing tasks. In such cases, not all students encounter the same errors and messages, and it is difficult to isolate the time spent interpreting messages and resolving errors from the time spent writing code. In this research, we explore the effects of presenting novices with compiler error messages designed using the most recent collection of published guidelines – specifically, more easily readable, short, positive messages containing resolution hints. To accurately determine the time and effort required to read and respond to the messages, we utilise a debugging task where all students are presented the same code and therefore encounter the same errors. We present results of a randomised controlled experiment ($n > 700$) which shows that, compared to standard error messages, the messages we tested resulted in significantly shorter debugging times and higher self-reported scores of message usefulness for students in the very early stages of learning a new language.

CCS CONCEPTS

• **Social and professional topics → CS1; Computing education; Computer science education;** • **Human-centered computing → Human computer interaction (HCI).**

KEYWORDS

compiler error messages; CS1; debugging; error message enhancement; novice programmers; programming error messages; readability

ACM Reference Format:
Paul Denny, James Prather, and Brett A. Becker. 2020. Error Message Readability and Novice Debugging Performance. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20), June 15–19, 2020, Trondheim, Norway.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3341525.3387384

1 INTRODUCTION

One of the challenges faced by any programmer new to a language is understanding the error messages produced by the compiler or interpreter. For students who are also new to programming, learning the syntax of a language alongside the principles of programming can be difficult, particularly when the messages they receive are confusing [17, 46]. Many educators will be familiar with the reality that certain error messages in the language they are teaching lead to particularly cryptic error messages which can be frustrating for students and hamper progress [5, 8].

In recent years there has been growing interest in understanding how students respond to various error messages and how those messages relate to underlying errors in code [7, 29, 37, 41, 44]. However, it is still widely accepted that for many languages, there is much room for improvement in the usefulness of error messages – particularly when concerning novice users [36].

A 2019 ITiCSE Working Group conducted a large-scale review of the research on programming error messages (PEMs) [6]. This report composed a list of design guidelines for improving text-based messages, based on those proposed by various researchers over the past 60 years. The guidelines were classified into ten categories, which included the following four: increase readability, reduce cognitive load, use a positive tone, and show solutions or hints. The Working Group report concluded with a call for additional research to empirically validate the guidelines for producing useful error messages: "Individual guidelines should be examined and then robustly tested to determine their effectiveness" [6, p. 204].

In this research we apply these guidelines to formulate new error messages for a select set of errors. We then measure the effect that these new messages have on students as they work through a simple debugging task. In particular, we explore how students perceive the usefulness of the new messages and how the new messages impact their debugging efforts. We answer the following research questions with respect to the newly formulated error messages:

RQ1: *Are the new error messages more readable, using traditional measures of readability, than the corresponding original compiler error messages?*

RQ2: *Do students read error messages and, if so, are the new messages perceived as more useful when debugging code in a new language, compared to the messages produced by the compiler?*

RQ3: *To what extent do the new messages impact debugging performance, in terms of time and effort?*

Denny, Prather & Becker (2020)

Example 1: Compiler error messages

Error Message Readability and Novice Debugging Performance

Paul Denny
University of Auckland
Auckland, New Zealand
paul@cs.auckland.ac.nz

James Prather
Abilene Christian University
Abilene, Texas, USA
james.prather@acu.edu

Brett A. Becker
University College Dublin
Dublin, Ireland
brett.becker@ucd.ie

ABSTRACT

It is well known that programming error messages can be notoriously difficult for novices to understand, hampering progress and leading to frustration. In response, researchers have explored various approaches for enhancing such messages, yet results from this active strand of research are currently mixed. Direct comparisons of results between studies is challenging as these typically investigate different kinds of message enhancements and report results using different metrics. In addition, many prior studies have involved code writing tasks. In such cases, not all students encounter the same errors and messages, and it is difficult to isolate the time spent interpreting messages and resolving errors from the time spent writing code. In this research, we explore the effects of presenting novices with compiler error messages designed using the most recent collection of published guidelines – specifically, more easily readable, short, positive messages containing resolution hints. To accurately determine the time and effort required to read and respond to the messages, we utilise a debugging task where all students are presented the same code and therefore encounter the same errors. We present results of a randomised controlled experiment ($n > 700$) which shows that, compared to standard error messages, the messages we tested resulted in significantly shorter debugging times and higher self-reported scores of message usefulness for students in the very early stages of learning a new language.

CCS CONCEPTS

• Social and professional topics → CS1; Computing education; Computer science education; • Human-centered computing → Human computer interaction (HCI).

KEYWORDS

compiler error messages; CS1; debugging; error message enhancement; novice programmers; programming error messages; readability

ACM Reference Format:
Paul Denny, James Prather, and Brett A. Becker. 2020. Error Message Readability and Novice Debugging Performance. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20), June 15–19, 2020, Trondheim, Norway.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3341525.3387384

ITiCSE '20, June 15–19, 2020, Trondheim, Norway

ACM ISBN 978-1-4503-6874-2/20/06…$15.00
https://doi.org/10.1145/3341525.3387384

1 INTRODUCTION

One of the challenges faced by any programmer new to a language is understanding the error messages produced by the compiler or interpreter. For students who are also new to programming, learning the syntax of a language alongside the principles of programming can be difficult, particularly when the messages they receive are confusing [17, 46]. Many educators will be familiar with the reality that certain error messages in the language they are teaching lead to particularly cryptic error messages which can be frustrating for students and hamper progress [5, 8].

In recent years there has been growing interest in understanding how students respond to various error messages and how those messages relate to underlying errors in code [7, 29, 37, 41, 44]. However, it is still widely accepted that for many languages, there is much room for improvement in the usefulness of error messages – particularly when concerning novice users [36].

A 2019 ITiCSE Working Group conducted a large-scale review of the research on programming error messages (PEMs) [6]. This report composed a list of design guidelines for improving text-based messages, based on those proposed by various researchers over the past 60 years. The guidelines were classified into ten categories, which included the following four: increase readability, reduce cognitive load, use a positive tone, and show solutions or hints. The Working Group report concluded with a call for additional research to empirically validate the guidelines for producing useful error messages: "Individual guidelines should be examined and then robustly tested to determine their effectiveness" [6, p. 204].

In this research we apply these guidelines to formulate new error messages for a select set of errors. We then measure the effect that these new messages have on students as they work through a simple debugging task. In particular, we explore how students perceive the usefulness of the new messages and how the new messages impact their debugging efforts. We answer the following research questions with respect to the newly formulated error messages:

RQ1: *Are the new error messages more readable, using traditional measures of readability, than the corresponding original compiler error messages?*

RQ2: *Do students read error messages and, if so, are the new messages perceived as more useful when debugging code in a new language, compared to the messages produced by the compiler?*

RQ3: *To what extent do the new messages impact debugging performance, in terms of time and effort?*

Denny, Prather & Becker (2020)

II. Background and Objectives

Manufacturer-supplied FORTRAN compilers normally provide rather efficient object code, provide flexible interaction with the operating systems, and have many sophisticated programming features. However, they are inadequate for the needs presented in the area of finding and correcting errors as quickly as possible. In many instances, the description of an error condition lacks resolution and offers the user little assistance in removing the error other than indicating the statement in which the error occurs. A more serious inadequacy is that many error descriptions are given in terms not understandable to a FORTRAN programmer.

DITRAN—a compiler emphasizing diagnostics
Moulton and Muller
Communications of the ACM
January 1967

Example 1: Compiler error messages

Error Message Readability and Novice Debugging Performance

Paul Denny
University of Auckland
Auckland, New Zealand
paul@cs.auckland.ac.nz

James Prather
Abilene Christian University
Abilene, Texas, USA
james.prather@acu.edu

Brett A. Becker
University College Dublin
Dublin, Ireland
brett.becker@ucd.ie

ABSTRACT

It is well known that programming error messages can be notoriously difficult for novices to understand, hampering progress and leading to frustration. In response, researchers have explored various approaches for enhancing such messages, yet results from this active strand of research are currently mixed. Direct comparisons of results between studies is challenging as these typically investigate different kinds of message enhancements and report results using different metrics. In addition, many prior studies have involved code writing tasks. In such cases, not all students encounter the same errors and messages, and it is difficult to isolate the time spent interpreting messages and resolving errors from the time spent writing code. In this research, we explore the effects of presenting novices with compiler error messages designed using the most recent collection of published guidelines – specifically, more easily readable, short, positive messages containing resolution hints. To accurately determine the time and effort required to read and respond to the messages, we utilise a debugging task where all students are presented the same code and therefore encounter the same errors. We present results of a randomised controlled experiment ($n > 700$) which shows that, compared to standard error messages, the messages we tested resulted in significantly shorter debugging times and higher self-reported scores of message usefulness for students in the very early stages of learning a new language.

CCS CONCEPTS

• Social and professional topics → CS1; Computing education; Computer science education; • Human-centered computing → Human computer interaction (HCI).

KEYWORDS

compiler error messages; CS1; debugging; error message enhancement; novice programmers; programming error messages; readability

ACM Reference Format:
Paul Denny, James Prather, and Brett A. Becker. 2020. Error Message Readability and Novice Debugging Performance. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20), June 15–19, 2020, Trondheim, Norway*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3341525.3387384

ITiCSE '20, June 15–19, 2020, Trondheim, Norway

ACM ISBN 978-1-4503-6874-2/20/06…$15.00
https://doi.org/10.1145/3341525.3387384

1 INTRODUCTION

One of the challenges faced by any programmer new to a language is understanding the error messages produced by the compiler or interpreter. For students who are also new to programming, learning the syntax of a language alongside the principles of programming can be difficult, particularly when the messages they receive are confusing [17, 46]. Many educators will be familiar with the reality that certain error messages in the language they are teaching lead to particularly cryptic error messages which can be frustrating for students and hamper progress [5, 8].

In recent years there has been growing interest in understanding how students respond to various error messages and how those messages relate to underlying errors in code [7, 29, 37, 41, 44]. However, it is still widely accepted that for many languages, there is much room for improvement in the usefulness of error messages – particularly when concerning novice users [36].

A 2019 ITiCSE Working Group conducted a large-scale review of the research on programming error messages (PEMs) [6]. This report composed a list of design guidelines for improving text-based messages, based on those proposed by various researchers over the past 60 years. The guidelines were classified into ten categories, which included the following four: increase readability, reduce cognitive load, use a positive tone, and show solutions or hints. The Working Group report concluded with a call for additional research to empirically validate the guidelines for producing useful error messages: "Individual guidelines should be examined and then robustly tested to determine their effectiveness" [6, p. 204].

In this research we apply these guidelines to formulate new error messages for a select set of errors. We then measure the effect that these new messages have on students as they work through a simple debugging task. In particular, we explore how students perceive the usefulness of the new messages and how the new messages impact their debugging efforts. We answer the following research questions with respect to the newly formulated error messages:

> **RQ1:** *Are the new error messages more readable, using traditional measures of readability, than the corresponding original compiler error messages?*
>
> **RQ2:** *Do students read error messages and, if so, are the new messages perceived as more useful when debugging code in a new language, compared to the messages produced by the compiler?*
>
> **RQ3:** *To what extent do the new messages impact debugging performance, in terms of time and effort?*

Denny, Prather & Becker (2020)

Yet, ask any experienced programmer about the quality of error messages in their programming environments, and you will often get an embarrassed laugh. In every environment, a mature programmer can usually point to at least a handful of favourite bad error responses. When they find out that the same environment is being used by novices, their laugh often hardens.

Marceau, Fisler & Krishnamurthi (2011)

Example 1: Compiler error messages

Error Message Readability and Novice Debugging Performance

Paul Denny
University of Auckland
Auckland, New Zealand
paul@cs.auckland.ac.nz

James Prather
Abilene Christian University
Abilene, Texas, USA
james.prather@acu.edu

Brett A. Becker
University College Dublin
Dublin, Ireland
brett.becker@ucd.ie

ABSTRACT

It is well known that programming error messages can be notoriously difficult for novices to understand, hampering progress and leading to frustration. In response, researchers have explored various approaches for enhancing such messages, yet results from this active strand of research are currently mixed. Direct comparisons of results between studies is challenging as these typically investigate different kinds of message enhancements and report results using different metrics. In addition, many prior studies have involved code writing tasks. In such cases, not all students encounter the same errors and messages, and it is difficult to isolate the time spent interpreting messages and resolving errors from the time spent writing code. In this research, we explore the effects of presenting novices with compiler error messages designed using the most recent collection of published guidelines – specifically, more easily readable, short, positive messages containing resolution hints. To accurately determine the time and effort required to read and respond to the messages, we utilise a debugging task where all students are presented the same code and therefore encounter the same errors. We present results of a randomised controlled experiment ($n > 700$) which shows that, compared to standard error messages, the messages we tested resulted in significantly shorter debugging times and higher self-reported scores of message usefulness for students in the very early stages of learning a new language.

CCS CONCEPTS

• Social and professional topics → CS1; Computing education; Computer science education; • Human-centered computing → Human computer interaction (HCI).

KEYWORDS

compiler error messages; CS1; debugging; error message enhancement; novice programmers; programming error messages; readability

ACM Reference Format:
Paul Denny, James Prather, and Brett A. Becker. 2020. Error Message Readability and Novice Debugging Performance. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20), June 15–19, 2020, Trondheim, Norway.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3341525.3387384

1 INTRODUCTION

One of the challenges faced by any programmer new to a language is understanding the error messages produced by the compiler or interpreter. For students who are also new to programming, learning the syntax of a language alongside the principles of programming can be difficult, particularly when the messages they receive are confusing [17, 46]. Many educators will be familiar with the reality that certain error messages in the language they are teaching lead to particularly cryptic error messages which can be frustrating for students and hamper progress [5, 8].

In recent years there has been growing interest in understanding how students respond to various error messages and how those messages relate to underlying errors in code [7, 29, 37, 41, 44]. However, it is still widely accepted that for many languages, there is much room for improvement in the usefulness of error messages – particularly when concerning novice users [36].

A 2019 ITiCSE Working Group conducted a large-scale review of the research on programming error messages (PEMs) [6]. This report composed a list of design guidelines for improving text-based messages, based on those proposed by various researchers over the past 60 years. The guidelines were classified into ten categories, which included the following four: increase readability, reduce cognitive load, use a positive tone, and show solutions or hints. The Working Group report concluded with a call for additional research to empirically validate the guidelines for producing useful error messages: "Individual guidelines should be examined and then robustly tested to determine their effectiveness" [6, p. 204].

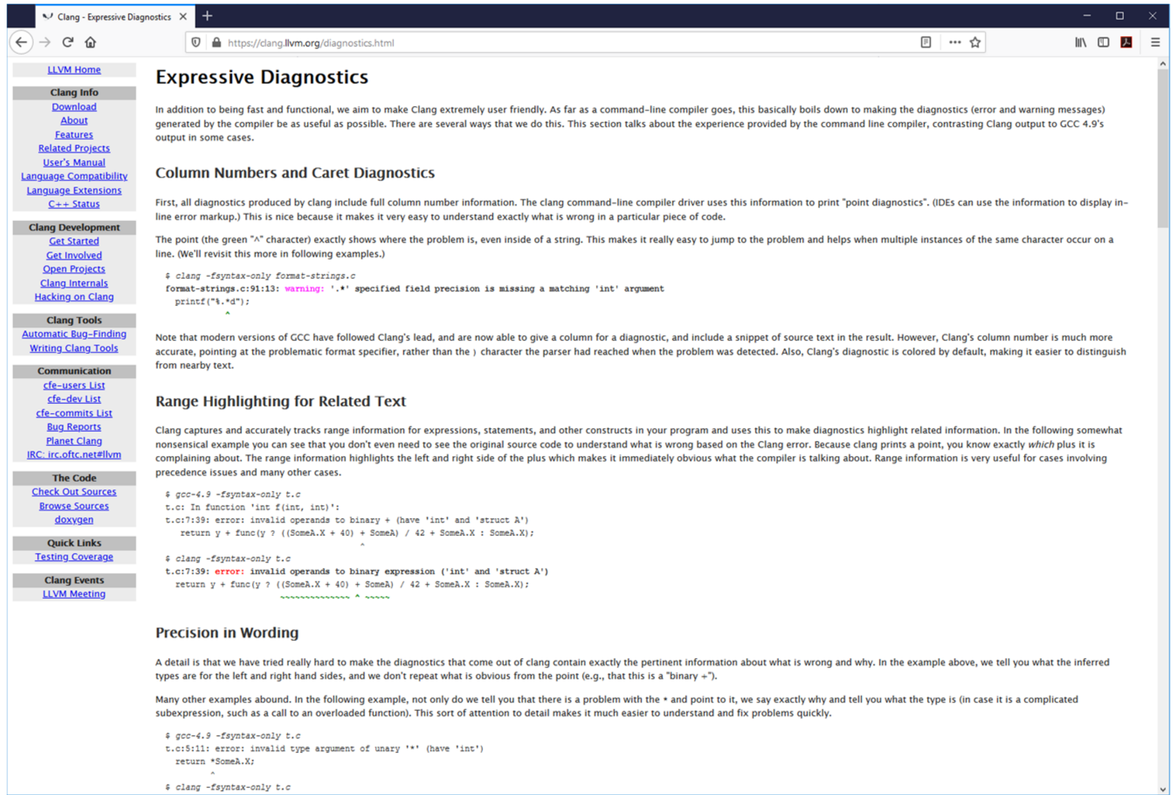
In this research we apply these guidelines to formulate new error messages for a select set of errors. We then measure the effect that these new messages have on students as they work through a simple debugging task. In particular, we explore how students perceive the usefulness of the new messages and how the new messages impact their debugging efforts. We answer the following research questions with respect to the newly formulated error messages:

RQ1: *Are the new error messages more readable, using traditional measures of readability, than the corresponding original compiler error messages?*

RQ2: *Do students read error messages and, if so, are the new messages perceived as more useful when debugging code in a new language, compared to the messages produced by the compiler?*

RQ3: *To what extent do the new messages impact debugging performance, in terms of time and effort?*

Denny, Prather & Becker (2020)



https://clang.llvm.org/diagnostics.html

Example 1: Compiler error messages

```
#include <studio.h>

#define CENTIMETERS_TO_FEET 0.0328
#define CENTIMETERS_TO_INCH 0.3937

int main(void)

    // Variables for converting metric to imperial
    int centimeters feet;
    double inches;

    // Read value into the variable centimeters
    scanf("%d", centimeters);

    feet = centimeters * CENTIMETERS_TO_FEET;
    inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
    printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);

    return 0;
}
```

Example 1: Compiler error messages

```
#include <studio.h>

#define CENTIMETERS_TO_FEET 0.0328
#define CENTIMETERS_TO_INCH 0.3937

int main(void)

    // Variables for converting metric to imperial
    int centimeters feet;
    double inches;

    // Read value into the variable centimeters
    scanf("%d", centimeters);

    feet = centimeters * CENTIMETERS_TO_FEET;
    inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
    printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);

    return 0;
}
```

1:10: fatal error: studio.h: No such file or directory

Example 1: Compiler error messages

```c
1   #include <studio.h>
2   
3   #define CENTIMETERS_TO_FEET 0.0328
4   #define CENTIMETERS_TO_INCH 0.3937
5   
6   int main(void)
7   
8       // Variables for converting metric to imperial
9       int centimeters feet;
10      double inches;
11  
12      // Read value into the variable centimeters
13      scanf("%d", centimeters);
14  
15      feet = centimeters * CENTIMETERS_TO_FEET;
16      inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
17      printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);
18  
19      return 0;
20  }
```

1:10: fatal error: studio.h: No such file or directory

Example 1: Compiler error messages

```
1   #include <studio.h>
2
3   #define CENTIMETERS_TO_FEET 0.0328
4   #define CENTIMETERS_TO_INCH 0.3937
5
6   int main(void)
7
8       // Variables for converting metric to imperial
9       int centimeters feet;
10      double inches;
11
12      // Read value into the variable centimeters
13      scanf("%d", centimeters);
14
15      feet = centimeters * CENTIMETERS_TO_FEET;
16      inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
17      printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);
18
19      return 0;
20  }
```

9:21: error: expected '=', ',', ';', 'asm' or '__attribute__' before 'feet'.

Example 1: Compiler error messages

```c
1  #include <studio.h>
2
3  #define CENTIMETERS_TO_FEET 0.0328
4  #define CENTIMETERS_TO_INCH 0.3937
5
6  int main(void)
7
8      // Variables for converting metric to imperial
9      int centimeters feet;
10     double inches;
11
12     // Read value into the variable centimeters
13     scanf("%d", centimeters);
14
15     feet = centimeters * CENTIMETERS_TO_FEET;
16     inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
17     printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);
18
19     return 0;
20 }
```

9:21: error: expected ‘=’, ‘,’, ‘;’, ‘asm’ or ‘__attribute__’ before ‘feet’.

Example 1: Compiler error messages

```
1   #include <studio.h>
2
3   #define CENTIMETERS_TO_FEET 0.0328
4   #define CENTIMETERS_TO_INCH 0.3937
5
6   int main(void)
7
8       // Variables for converting metric to imperial
9       int centimeters feet;
10      double inches;
11
12      // Read value into the variable centimeters
13      scanf("%d", centimeters);
14
15      feet = centimeters * CENTIMETERS_TO_FEET;
16      inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
17      printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);
18
19      return 0;
20  }
```

13:5: error: expected declaration specifiers before ‘scanf’

Example 1: Compiler error messages

```c
1  #include <studio.h>
2
3  #define CENTIMETERS_TO_FEET 0.0328
4  #define CENTIMETERS_TO_INCH 0.3937
5
6  int main(void)
7
8      // Variables for converting metric to imperial
9      int centimeters feet;
10     double inches;
11
12     // Read value into the variable centimeters
13     scanf("%d", centimeters);
14
15     feet = centimeters * CENTIMETERS_TO_FEET;
16     inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
17     printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);
18
19     return 0;
20 }
```

13:5: error: expected declaration specifiers before ‘scanf’

Example 1: Compiler error messages

```
1  #include <studio.h>
2
3  #define CENTIMETERS_TO_FEET 0.0328
4  #define CENTIMETERS_TO_INCH 0.3937
5
6  int main(void)
7
8      // Variables for converting metric to imperial
9      int centimeters feet;
10     double inches;
11
12     // Read value into the variable centimeters
13     scanf("%d", centimeters);
14
15     feet = centimeters * CENTIMETERS_TO_FEET;
16     inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
17     printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);
18
19     return 0;
20 }
```

13:13: error: format ‘%d’ expects argument of type ‘int *’, but argument 2 has type ‘int’.

Example 1: Compiler error messages

```c
1  #include <studio.h>
2
3  #define CENTIMETERS_TO_FEET 0.0328
4  #define CENTIMETERS_TO_INCH 0.3937
5
6  int main(void)
7
8      // Variables for converting metric to imperial
9      int centimeters feet;
10     double inches;
11
12     // Read value into the variable centimeters
13     scanf("%d", centimeters);
14
15     feet = centimeters * CENTIMETERS_TO_FEET;
16     inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
17     printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);
18
19     return 0;
20 }
```

13:13: error: format ‘%d’ expects argument of type ‘int *’, but argument 2 has type ‘int’.

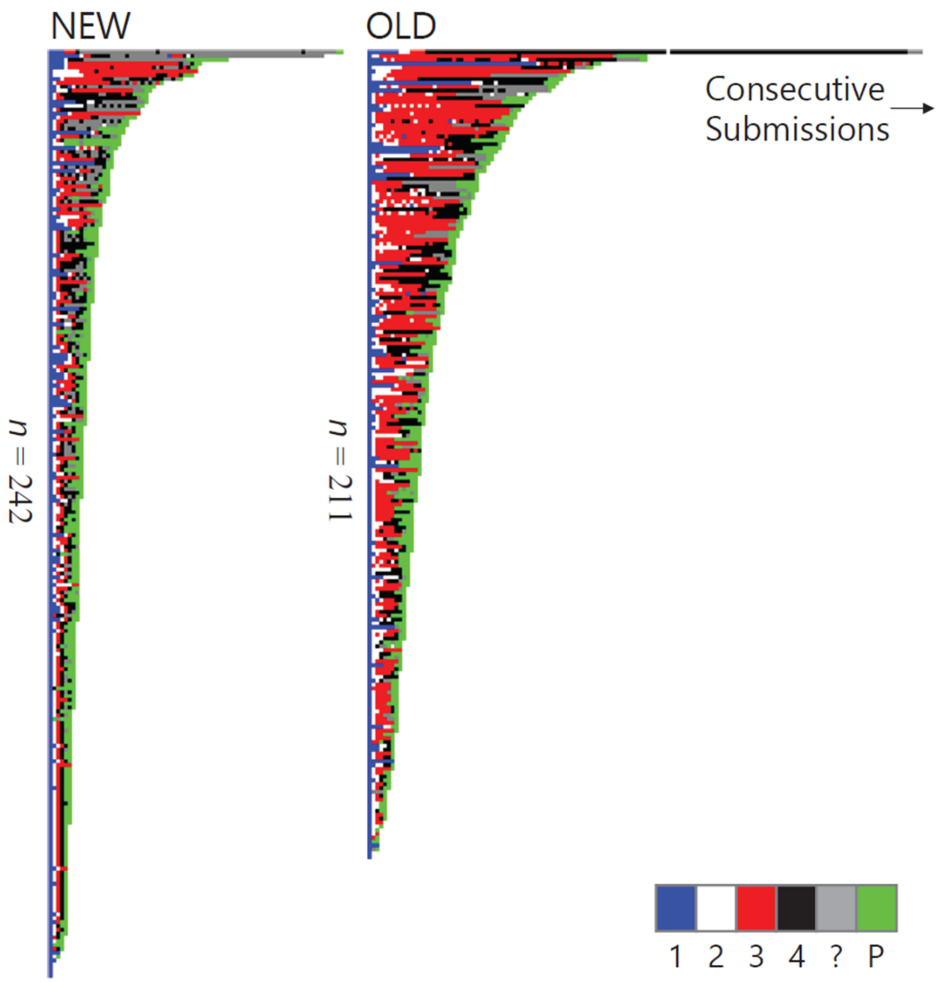
Example 1: Compiler error messages

```
#include <studio.h>

#define CENTIMETERS_TO_FEET 0.0328
#define CENTIMETERS_TO_INCH 0.3937

int main(void)

    // Variables for converting metric to imperial
    int centimeters feet;
    double inches;

    // Read value into the variable centimeters
    scanf("%d", centimeters);

    feet = centimeters * CENTIMETERS_TO_FEET;
    inches = (centimeters - feet / CENTIMETERS_TO_FEET) * CENTIMETERS_TO_INCH;
    printf("%d centimeters is %d feet and %.2f inches\n", centimeters, feet, inches);

    return 0;
}
```

Could we provide more useful error messages?

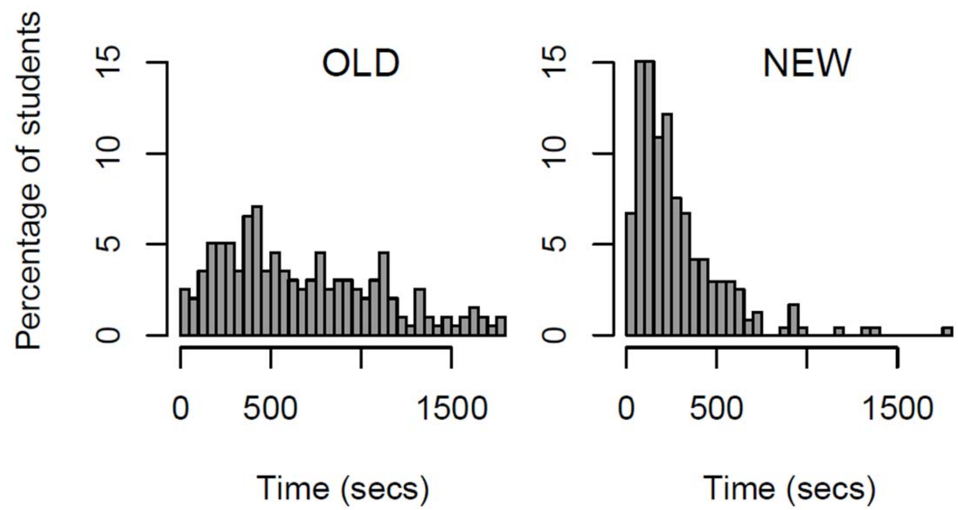
What sort of difference would that make?

Example 1: Compiler error messages



Could we provide more useful error messages?

What sort of difference would that make?



Example 2: Influencing (positive) behaviours

CHI 2018 Honourable Mention
CHI 2018, April 21–26, 2018, Montréal, QC, Canada

Empirical Support for a Causal Relationship Between Gamification and Learning Outcomes

Paul Denny
University of Auckland
Auckland, New Zealand
paul@cs.auckland.ac.nz

Fiona McDonald, Ruth Empson, Philip Kelly
University of Otago
Dunedin, New Zealand
{fiona.mcdonald, ruth.empson, philip.kelly}@otago.ac.nz

Andrew Petersen
University of Toronto
Mississauga, Canada
andrew.petersen@utoronto.ca

ABSTRACT

Preparing for exams is an important yet stressful time for many students. Self-testing is known to be an effective preparation strategy, yet some students lack motivation to engage or persist in self-testing activities. Adding game elements to a platform supporting self-testing may increase engagement and, by extension, exam performance. We conduct a randomized controlled experiment (n=701) comparing the effect of two game elements – a points system and a badge system – used individually and in combination.

We find that the badge system elicits significantly higher levels of voluntary self-testing activity and this effect is particularly pronounced amongst a relatively small cohort. Importantly, this increased activity translates to a significant improvement in exam scores. Our data supports a causal relationship between gamification and learning outcomes, mediated by self-testing behavior. This provides empirical support for Landers' theory of gamified learning when the gamified activity is conducted prior to measuring learning outcomes.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous; K.3.0 Computers and Education: General

Author Keywords

gamification; points; badges; self-testing; PeerWise

INTRODUCTION

A growing number of online platforms are incorporating game-like elements to motivate users and generate higher levels of activity. Commonly referred to as "gamification," this approach employs elements that are typically seen in games in non-game contexts [18]. Educational tools have followed this trend, with many including features such as points [10], leaderboards [4], levels [43] and virtual achievements [15].

This raises the question, "Can gamification positively impact student behavior and learning outcomes?"

In a comprehensive review of the literature, Hamari et al. report that three particular elements: points, leaderboards and badges are the most commonly used in empirical studies of gamification [28]. Their review concluded that most published gamification studies reported some positive effects, but they identified a number of methodological limitations that may have contributed to varying results. These limitations included small sample sizes, lack of control groups and very short experiment timeframes. A fourth limitation was that multiple game elements were often investigated in combination, but not individually, making it impossible to establish whether individual elements had measurable effects.

In this work we investigate two of the most common gamification elements, points and badges, as used in an online learning tool. Our context is a large first-year anatomy and physiology course (701 participants), where we investigate student engagement with the tool over an entire 15 week semester and relate engagement to subsequent exam performance. We examine the effects of the game elements both individually and in combination, relative to a control group.

We explore two related research questions. Our primary question tests the hypothesis that gamifying an online study tool will have a causal effect on subsequent exam performance. Landers' theory of gamified learning provides strong theoretical support for this hypothesis [34]. Our secondary research question tests the hypothesis that a combination of game elements will have a greater effect on student behavior than either element used on its own. We measure the individual effects of our implemented points and badge systems, and we determine if their simultaneous use is beneficial in our context.

BACKGROUND

Education is an increasingly common application area for gamification [2, 33, 51]. This has been driven by the potential for gamification to address challenges around student motivation and to positively impact learning [8, 36]. This latter outcome is of particular importance in educational contexts. The relationship between gamification and learning outcomes may be mediated by behaviors, such as time-on-task, that

Denny, McDonald, Empson, Kelly & Petersen (2018)

Example 2: Influencing (positive) behaviours

CHI 2018 Honourable Mention   CHI 2018, April 21–26, 2018, Montréal, QC, Canada

Empirical Support for a Causal Relationship Between Gamification and Learning Outcomes

Paul Denny
University of Auckland
Auckland, New Zealand
paul@cs.auckland.ac.nz

Fiona McDonald, Ruth Empson, Philip Kelly
University of Otago
Dunedin, New Zealand
{fiona.mcdonald, ruth.empson, philip.kelly}@otago.ac.nz

Andrew Petersen
University of Toronto
Mississauga, Canada
andrew.petersen@utoronto.ca

ABSTRACT
Preparing for exams is an important yet stressful time for many students. Self-testing is known to be an effective preparation strategy, yet some students lack motivation to engage or persist in self-testing activities. Adding game elements to a platform supporting self-testing may increase engagement and, by extension, exam performance. We conduct a randomized controlled experiment (n=701) comparing the effect of two game elements – a points system and a badge system – used individually and in combination.

We find that the badge system elicits significantly higher levels of voluntary self-testing activity and this effect is particularly pronounced amongst a relatively small cohort. Importantly, this increased activity translates to a significant improvement in exam scores. Our data supports a causal relationship between gamification and learning outcomes, mediated by self-testing behavior. This provides empirical support for Landers' theory of gamified learning when the gamified activity is conducted prior to measuring learning outcomes.

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous; K.3.0 Computers and Education: General

Author Keywords
gamification; points; badges; self-testing; PeerWise

INTRODUCTION
A growing number of online platforms are incorporating game-like elements to motivate users and generate higher levels of activity. Commonly referred to as "gamification," this approach employs elements that are typically seen in games in non-game contexts [18]. Educational tools have followed this trend, with many including features such as points [10], leaderboards [4], levels [43] and virtual achievements [15].

This raises the question, "Can gamification positively impact student behavior and learning outcomes?"

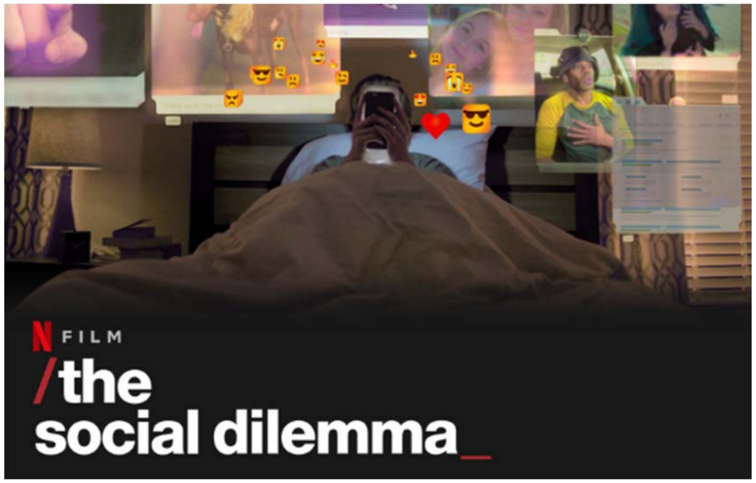
In a comprehensive review of the literature, Hamari et al. report that three particular elements: points, leaderboards and badges are the most commonly used in empirical studies of gamification [28]. Their review concluded that most published gamification studies reported some positive effects, but they identified a number of methodological limitations that may have contributed to varying results. These limitations included small sample sizes, lack of control groups and very short experiment timeframes. A fourth limitation was that multiple game elements were often investigated in combination, but not individually, making it impossible to establish whether individual elements had measurable effects.

In this work we investigate two of the most common gamification elements, points and badges, as used in an online learning tool. Our context is a large first-year anatomy and physiology course (701 participants), where we investigate student engagement with the tool over an entire 15 week semester and relate engagement to subsequent exam performance. We examine the effects of the game elements both individually and in combination, relative to a control group.

We explore two related research questions. Our primary question tests the hypothesis that gamifying an online study tool will have a causal effect on subsequent exam performance. Landers' theory of gamified learning provides strong theoretical support for this hypothesis [34]. Our secondary research question tests the hypothesis that a combination of game elements will have a greater effect on student behavior than either element used on its own. We measure the individual effects of our implemented points and badge systems, and we determine if their simultaneous use is beneficial in our context.

BACKGROUND
Education is an increasingly common application area for gamification [2, 33, 51]. This has been driven by the potential for gamification to address challenges around student motivation and to positively impact learning [8, 36]. This latter outcome is of particular importance in educational contexts. The relationship between gamification and learning outcomes may be mediated by behaviors, such as time-on-task, that

Denny, McDonald, Empson, Kelly & Petersen (2018)



You're probably a kick-ass developer... but are you a secure developer? Try our gamified challenges, climb the leaderboard, and win a FREE T-shirt! (securecodewarrior.com)
promoted by SecCodeWarrior
promoted save give award report

Example 2: Influencing (positive) behaviours

- Or, avoiding (negative) behaviours

Paper Session: CS1 Metacognitive CompEd '19, May 17–19, 2019, Chengdu, Sichuan, China

Can Mobile Gaming Psychology Be Used to Improve Time Management on Programming Assignments?

Michael S. Irwin and Stephen H. Edwards
Department of Computer Science
Virginia Tech
Blacksburg, VA, USA
mikesir@vt.edu, edwards@cs.vt.edu

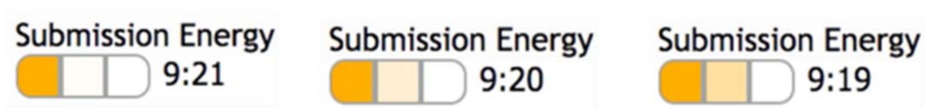


Figure 1: Submission energy bar, showing countdown and the animated fading of the next available unit being regenerated.

Irwin & Edwards (2019)

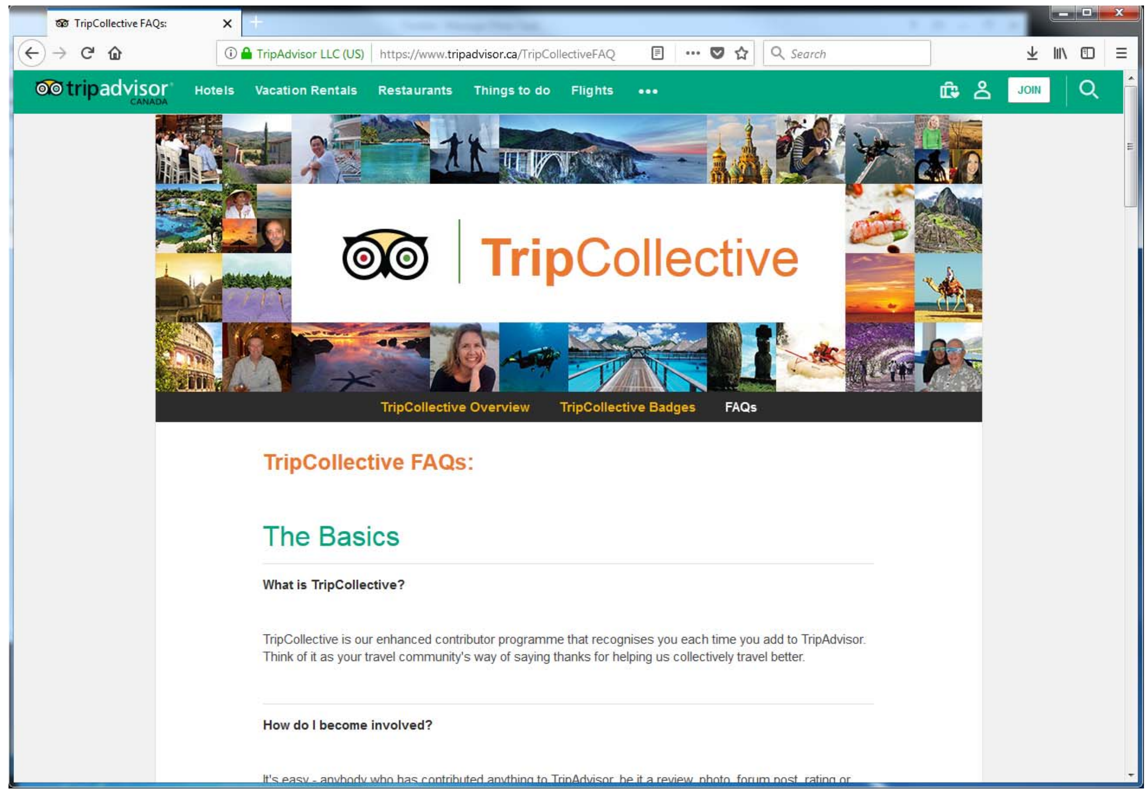
Example 2: Influencing (positive) behaviours

The Programme

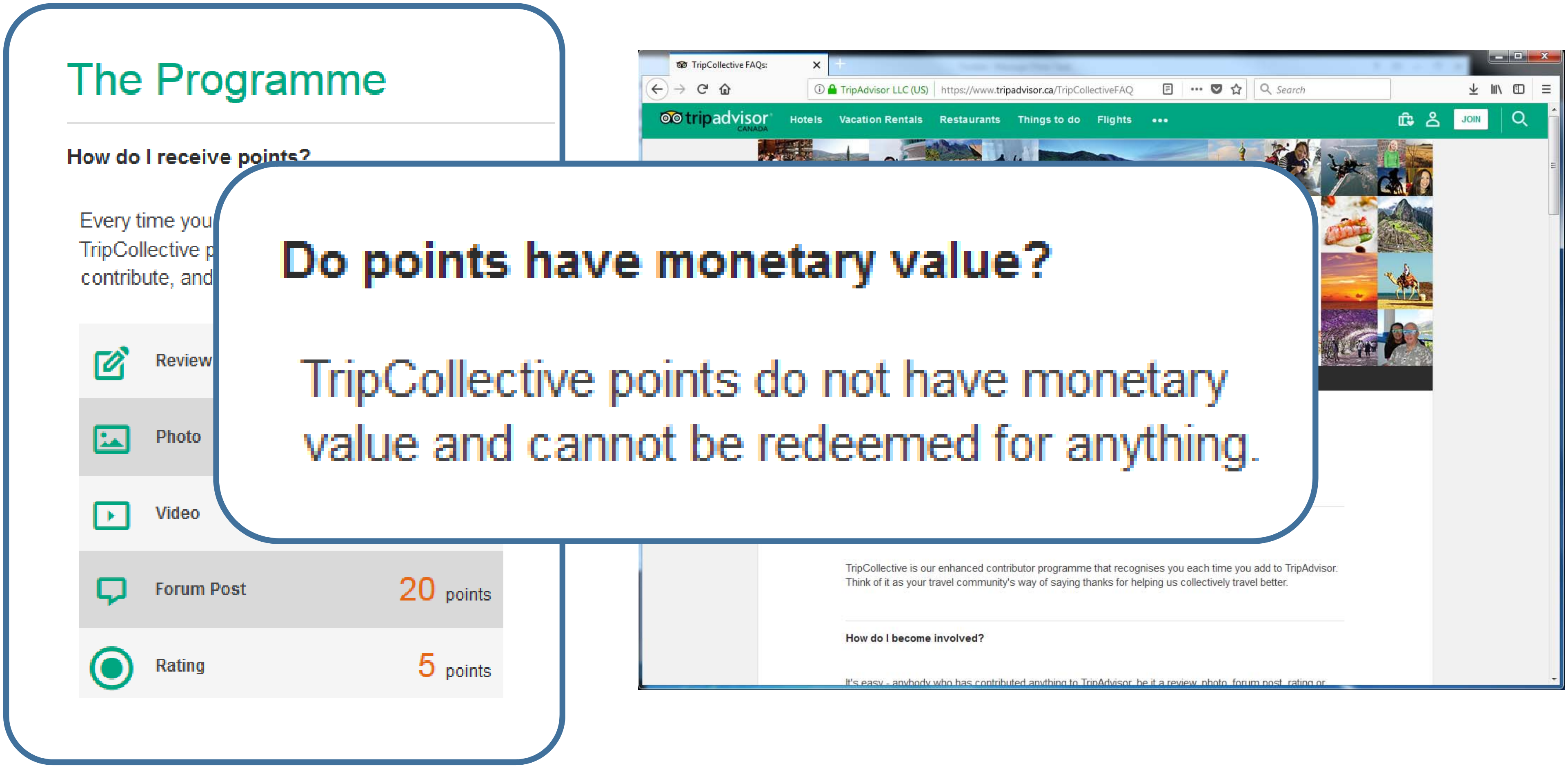
How do I receive points?

Every time you contribute to TripAdvisor, you receive TripCollective points. Here's a list of what you can contribute, and how much it's worth.

Review	100 points
Photo	30 points
Video	30 points
Forum Post	20 points
Rating	5 points



Example 2: Influencing (positive) behaviours



Example 2: Influencing (positive) behaviours

“Empirical research on the *effectiveness* of incorporating game elements in learning environments is still scarce”

[Dicheva et al.; 2015]

Dicheva, Dichev, Agre, Angelova. 2015. "**Gamification in Education: A Systematic Mapping Study**", Journal of Educational Technology & Society, Vol. 18, No. 3 (July 2015), pp. 75-88

Example 2: Influencing (positive) behaviours

Topical Article

SOCIETY FOR THE TEACHING OF PSYCHOLOGY

Teaching of Psychology
2019, Vol. 46(2) 121-126

DOI: 10.1177/0098628319834174
journals.sagepub.com/home/top
SAGE

Generation and Retrieval Practice Effects in the Classroom Using PeerWise

Matthew R. Kelley[1], Elizabeth K. Chapman-Orr[2], Susanna Calkins[3], and Robert J. Lemke[4]

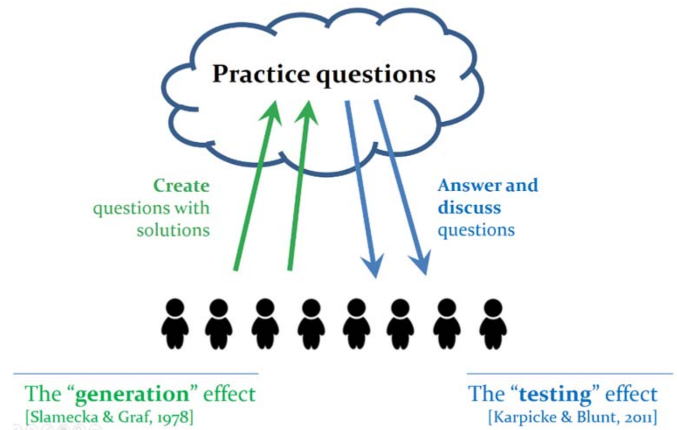
Abstract

The present study explored the generation and retrieval practice effects within a college classroom using a free, online tool called PeerWise (PW). PW allows students to create their own multiple-choice questions, share them with peers, and answer the shared questions written by their peers. Forty students from two sections of an upper level cognitive psychology course authored and answered multiple-choice questions as part of a semester-long assignment. Analyses showed reliable generation and retrieval practice effects following PW usage, along with a significant improvement in exam performance.

Keywords

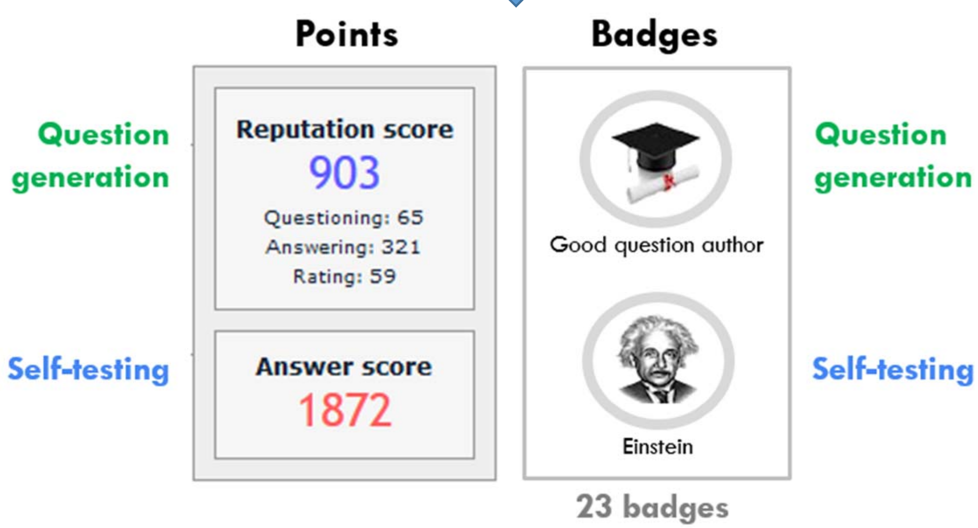
generation effect, retrieval practice, PeerWise

Kelley, Chapman-Orr, Calkins, Lemke. **Generation and Retrieval Practice Effects in the Classroom Using PeerWise,** Teaching of Psychology, March 1, 2019.

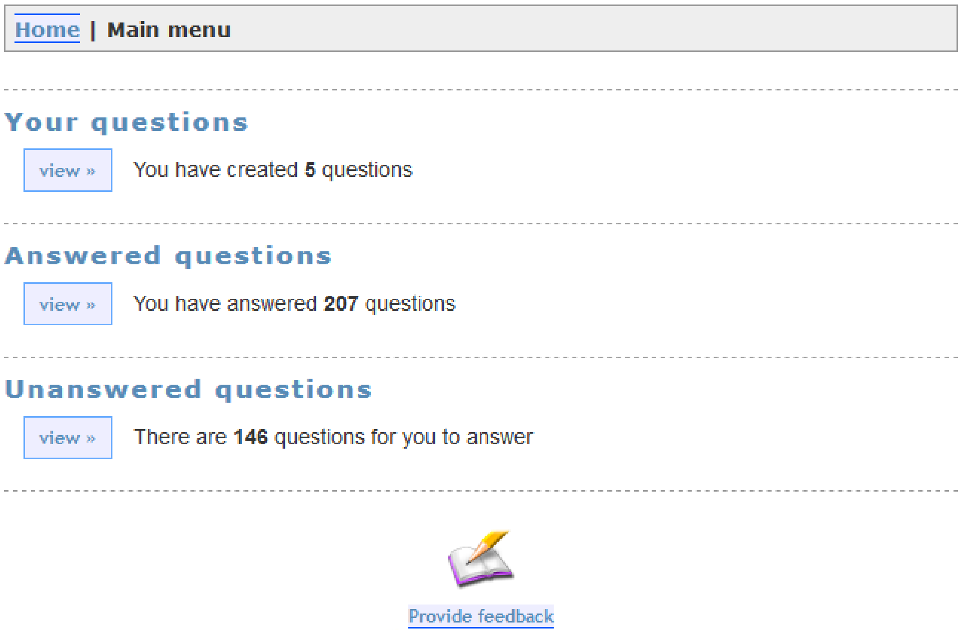


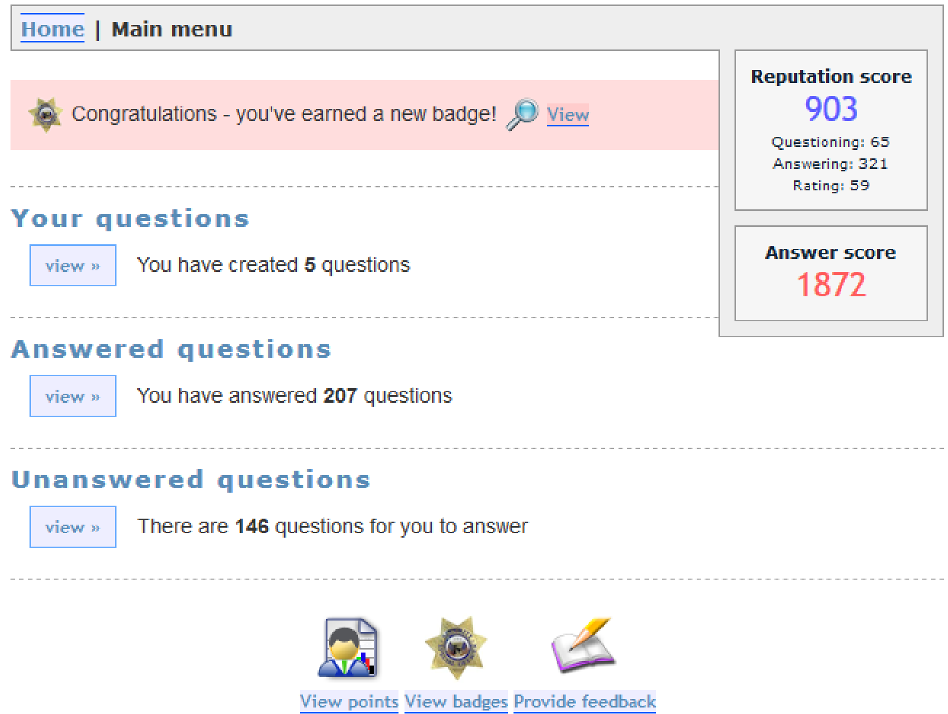
Example 2: Influencing (positive) behaviours

Desired behavior	Example reward
Rate questions early and fairly	Reputation score (rating component)
Answer questions correctly	Answer score
Spaced practice sessions	Commitment badge
Create good questions	Good question author badge
....	....



Example 2: Influencing (positive) behaviours





control vs. game

Denny, McDonald, Empson, Kelly, Petersen. 2018. **Empirical Support for a Causal Relationship Between Gamification and Learning Outcomes.** In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). ACM, New York, NY, USA

Example 2: Influencing (positive) behaviours

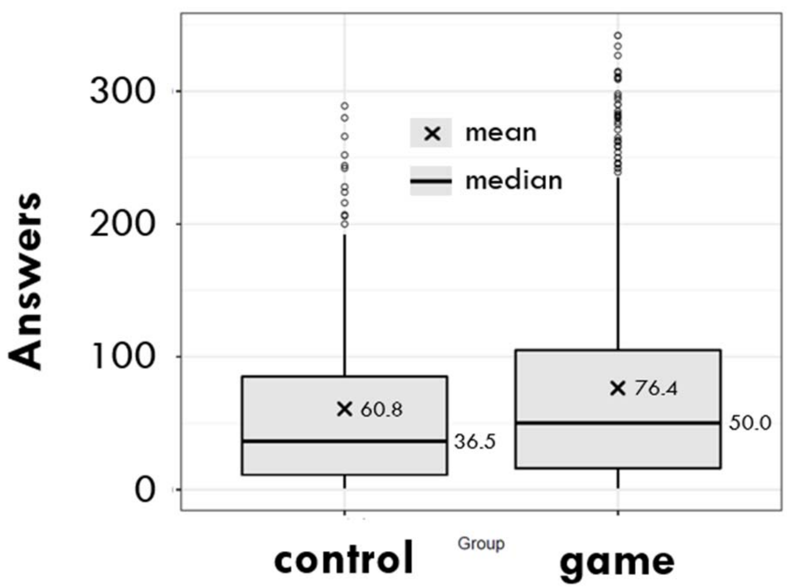
“Personally I tried really hard to get the ‘Leader’ badge, where I had to gain at least one follower! This was really motivating, and made me think more carefully and creatively when writing a question.

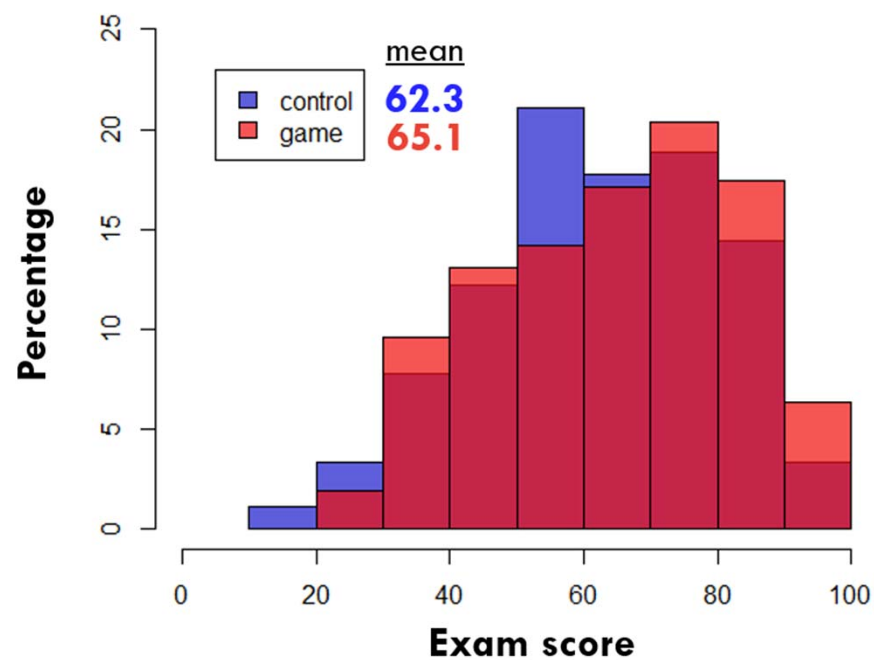
“I didn't think I was ‘badge’ type of person, but I did enjoy getting badges (I was the first one to get the ‘obsessed badge’ - yay!). It did help motivate me to do extra and in doing so, I believe I have learnt more effectively.”

Example 2: Influencing (positive) behaviours

Game elements caused:

- twice as many students to create questions (6.7% vs. 11.5%)
- nearly 40% increase in answering activity





37% increase in median number of answers
(p = .016)

4.5% increase in mean exam score
(p = .038)

Summary

- The goal of Computing Education research is to help students learn computer science more effectively
- The goal of Learning Technology research is the same, but applies more broadly across disciplines
- These are interesting areas of research, to which a range of computer science skills can be applied, and with the potential of large impact
- Our School’s graduate course is COMPSCI 747!