University of Roehampton London

Computação nas escolas inglesas: lições aprendidas

Miles Berry

8 de setembro de 2021

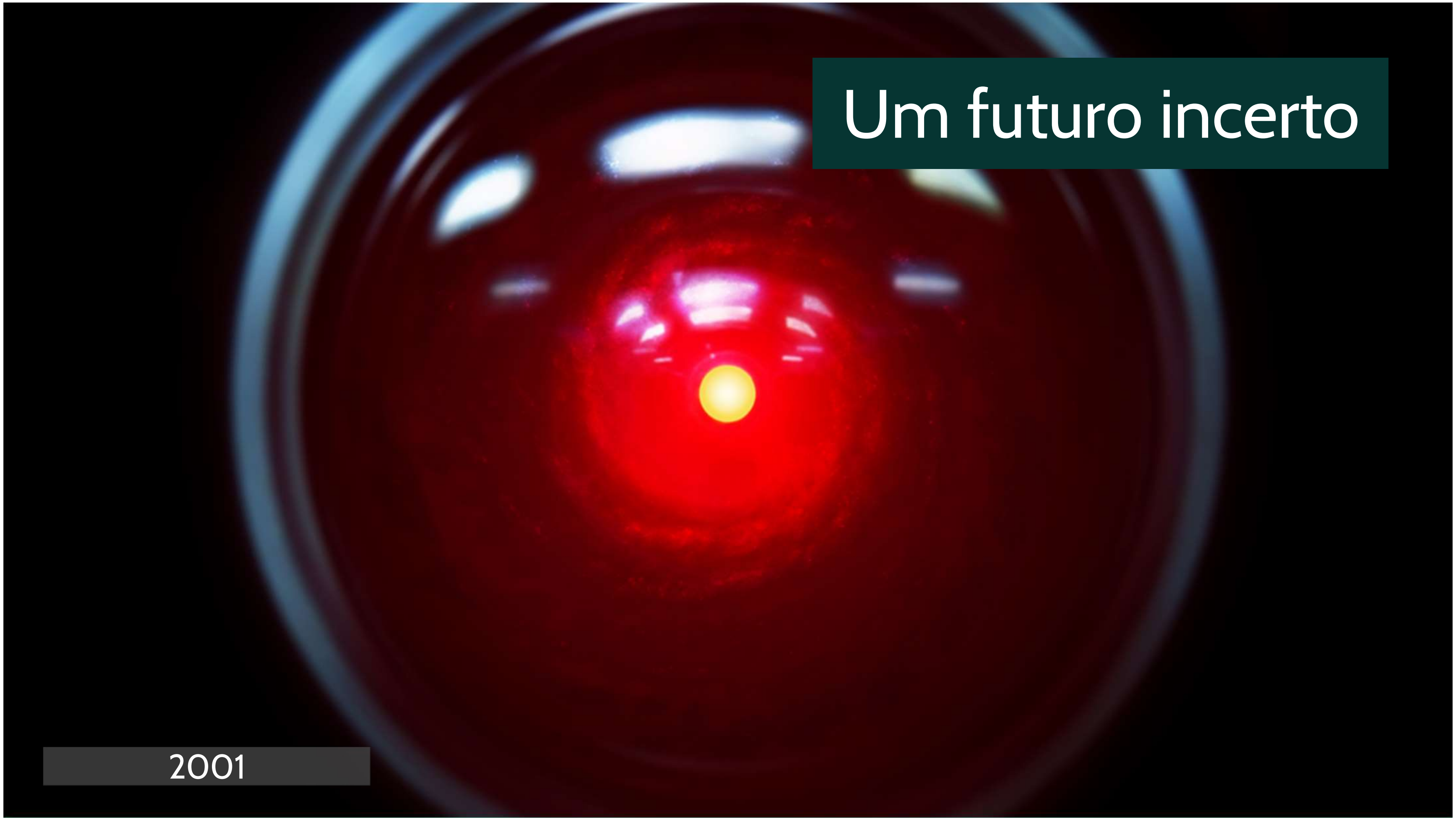
ERTE5

Para os Açores

Arranjar um emprego


Pergunto-me como é que isto funciona?
SAMSUNG
chrome


Uma educação liberal no terceiro milénio
Herts Music Service


Um futuro incerto
2001

Ferramentas para pensar com

Papert, Mindstorms

Inclusão


qbmaze


qbmaze

CS, IT, DL

Fundações

Candidaturas

Implicações

Pensamento computacional

Criatividade

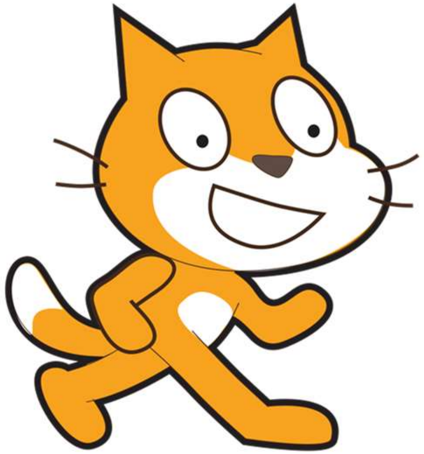
Compreender o mundo

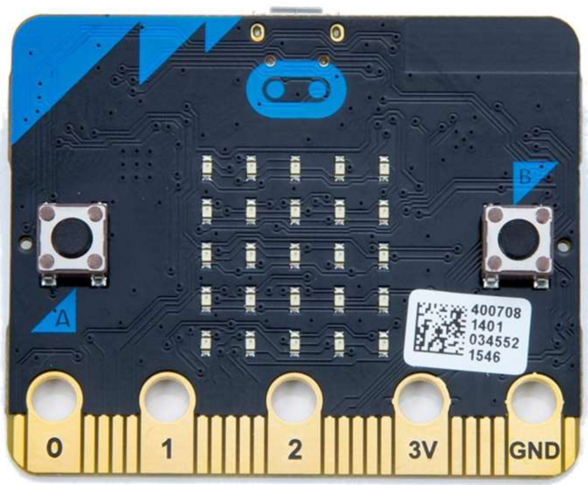
Mudar o mundo

Pensamento computacional

Lógica	Retoque
Algoritmos	Criação
Decomposição	Depurando
Padrões	Perseverar
Abstração	Colaboração


GO




A
B
400708
1401
034552
1546
0
1
2
3V
GND

Para fazer...

Objetivos e âmbito

Objetivos globais de aprendizagem por vertente e nota

Planeamento e recursos para o Grau 1

Desenvolvimento profissional

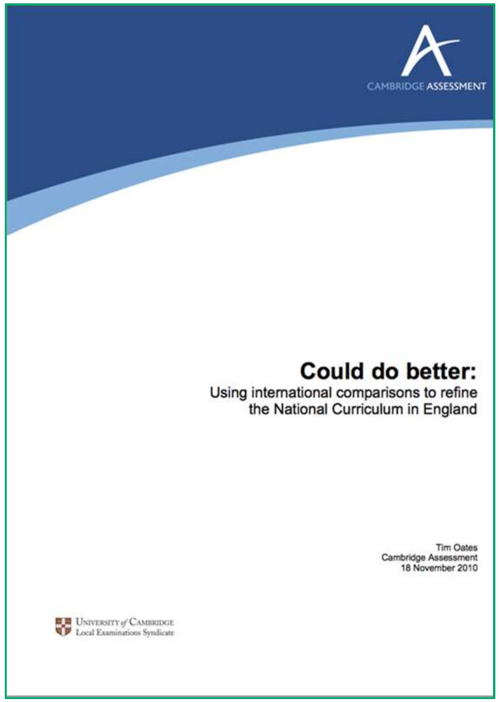
Avaliação

	CS	IT	DL
1			
2			
3			
4			
5			
6			

Da Inglaterra

Coerência Curricular

O termo «coerência» é um termo técnico muito preciso: um currículo nacional deve dispor de conteúdos organizados por uma ordem que se baseie em elementos de prova associados à progressão relacionada com a idade, e todos os elementos do sistema (conteúdo, avaliação, pedagogia, formação de professores, materiais pedagógicos, incentivos e motoristas, etc.) devem alinhar-se e agir de forma concertada para a entrega de bens públicos.



Oates, 2010

Conteúdo

O currículo de computação

> "Substituímos o antigo currículo de tecnologias de informação e comunicação por um novo currículo de computação, com a ajuda do Google, Facebook e alguns dos mais brilhantes cientistas de computação da Grã-Bretanha."

Gove, 2013

Quatro objetivos

pode compreender e aplicar os princípios e conceitos fundamentais da ciência da computação, incluindo abstração, lógica, algoritmos e representação de dados

pode analisar problemas em termos computacionais, e ter experiência prática repetida de escrever programas de computador, a fim de resolver tais problemas

pode avaliar e aplicar tecnologias de informação, incluindo novas ou desconhecidas, analíticas para resolver problemas

são responsáveis, competentes, confiantes e criativos utilizadores de tecnologias de informação e comunicação

Antes das 5

Brincar e explorar

Percebam que as suas ações têm um efeito no mundo

Planeie e pense com antecedência sobre como eles vão explorar ou brincar com objetos

Aprendizagem ativa

Começam a prever sequências porque conhecem rotinas

Mostrar comportamento direcionado para o objetivo

Use uma série de estratégias para alcançar um objetivo que eles próprios definiram

Começar a corrigir os seus próprios erros.

Continua a tentar quando as coisas ficam difíceis.

Criar e pensar criticamente

Participe de uma simples peça de fingimento

Reveja os progressos enquanto tentam atingir um objetivo. Vê se estão a sair-se bem.

Resolver problemas do mundo real

Use a peça de fingimento para pensar além do "aqui e agora"

Concentre-se em alcançar algo que seja importante para eles.

Development Matters, 2020

De 5 a 16

Computing

Purpose of study

A high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world. Computing has deep links with mathematics, science, and design and technology, and provides insights into both natural and artificial systems. The core of computing is computer science, in which pupils are taught the principles of information and computation, how digital systems work, and how to put this knowledge to use through programming. Building on this knowledge and understanding, pupils are equipped to use information technology to create programs, systems and a range of content. Computing also ensures that pupils become digitally literate – able to use, and express themselves and develop their ideas through, information and communication technology – at a level suitable for the future workplace and as active participants in a digital world.

Aims

The national curriculum for computing aims to ensure that all pupils:

- can understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation
- can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems
- can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems
- are responsible, competent, confident and creative users of information and communication technology.

Attainment targets

By the end of each key stage, pupils are expected to know, apply and understand the matters, skills and processes specified in the relevant programme of study.

Schools are not required by law to teach the example content in [square brackets].

Subject content

Key stage 1

Pupils should be taught to:

- understand what algorithms are; how they are implemented as programs on digital devices; and that programs execute by following precise and unambiguous instructions
- create and debug simple programs
- use logical reasoning to predict the behaviour of simple programs
- use technology purposefully to create, organise, store, manipulate and retrieve digital content
- recognise common uses of information technology beyond school
- use technology safely and respectfully, keeping personal information private; identify where to go for help and support when they have concerns about content or contact on the internet or other online technologies.

Key stage 2

Pupils should be taught to:

- design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts
- use sequence, selection, and repetition in programs; work with variables and various forms of input and output
- use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs
- understand computer networks including the internet; how they can provide multiple services, such as the world wide web; and the opportunities they offer for communication and collaboration
- use search technologies effectively, appreciate how results are selected and ranked, and be discerning in evaluating digital content
- select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information
- use technology safely, respectfully and responsibly; recognise acceptable/unacceptable behaviour; identify a range of ways to report concerns about content and contact.

Key stage 3

Pupils should be taught to:

- design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- understand several key algorithms that reflect computational thinking [for example, ones for sorting and searching]; use logical reasoning to compare the utility of alternative algorithms for the same problem
- use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures [for example, lists, tables or arrays]; design and develop modular programs that use procedures or functions
- understand simple Boolean logic [for example, AND, OR and NOT] and some of its uses in circuits and programming; understand how numbers can be represented in binary, and be able to carry out simple operations on binary numbers [for example, binary addition, and conversion between binary and decimal]
- understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems
- understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits
- undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users
- create, re-use, revise and re-purpose digital artefacts for a given audience, with attention to trustworthiness, design and usability
- understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct and know how to report concerns.

Key stage 4

All pupils must have the opportunity to study aspects of information technology and computer science at sufficient depth to allow them to progress to higher levels of study or to a professional career.

All pupils should be taught to:

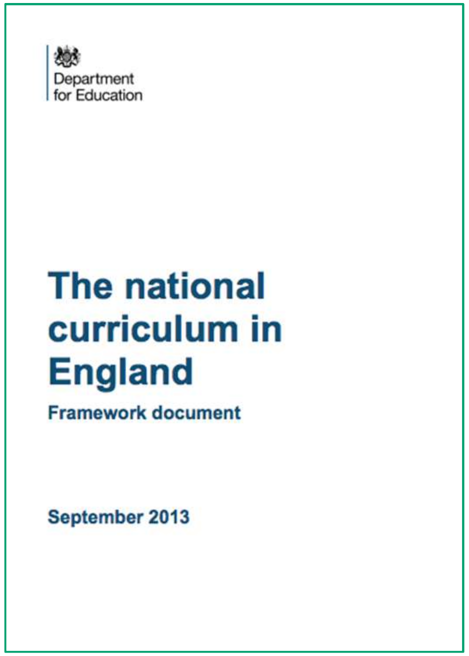
- develop their capability, creativity and knowledge in computer science, digital media and information technology
- develop and apply their analytic, problem-solving, design, and computational thinking skills
- understand how changes in technology affect safety, including new ways to protect their online privacy and identity, and how to identify and report a range of concerns.

DfE, 2013

Avaliação

Alvo de realização

Até ao final de cada fase-chave, espera-se que os alunos conheçam, apliquem e compreendam as matérias, competências e processos especificados no programa de estudo relevante.


Department for Education

The national curriculum in England

Framework document

September 2013

DfE, 2013

Assessing attainment in computing – a national curriculum framework

	CS	IT	DL
KS1 - 1	Understand what algorithms are Create simple programs	Use technology purposefully to create digital content Use technology purposefully to store digital content Use technology purposefully to retrieve digital content	Use technology safely Keep personal information private Recognise common uses of information technology beyond school
KS1 - 2	Understand that algorithms are implemented as programs on digital devices Understand that programs execute by following precise and unambiguous instructions Debug simple programs Use logical reasoning to predict the behaviour of simple programs	Use technology purposefully to organise digital content Use technology purposefully to manipulate digital content	Use technology respectfully Identify where to go for help and support when they have concerns about content or contact on the internet or other online technologies
KS2 -3	Write programs that accomplish specific goals Use sequence in programs Work with various forms of input Work with various forms of output	Use search technologies effectively Use a variety of software to accomplish given goals Collect information Design and create content Present information	Use technology responsibly Identify a range of ways to report concerns about contact
KS2 - 4	Design programs that accomplish specific goals Design and create program Debug programs that accomplish specific goals Use repetition in programs Control or simulate physical systems Use logical reasoning to detect and correct errors in programs Understand how computer networks can provide multiple services, such as the world wide web Appreciate how search results are selected	Select a variety of software to accomplish given goals Select, use and combine internet services Analyse information Evaluate information Collect data Present data	Understand the opportunities computer networks offer for communication Identify a range of ways to report concerns about content Recognize acceptable / unacceptable behaviour
KS2 -5	Solve problems by decomposing them into smaller parts Use selection in programs Work with variables Use logical reasoning to explain how some simple algorithms work Use logical reasoning to detect and correct errors in algorithms Understand computer networks including the internet Appreciate how search results are ranked	Combine a variety of software to accomplish given goals Select use and combine software on a range of digital devices Analyse data Evaluate data Design and create systems	Understand the opportunities computer networks offer for collaboration Be discerning in evaluating digital content
KS3 - 6	Use computational abstractions Model state of real world problems Use a programming language to solve computational problems Understand simple Boolean logic Understand how numbers can be represented in binary Understand the hardware components that make up computer systems Understand how text can be represented digitally in the form of binary digits Understand how pictures can be represented digitally in the form of binary digits	Undertake creative projects with challenging goals Use multiple applications [Work with] applications across a range of devices Collect data	Understand a range of ways to use technology respectfully Recognise inappropriate content Recognise inappropriate contact Recognise inappropriate conduct Know how to report concerns Reuse digital artefacts for a given audience Attend to usability of digital artefacts Understand a range of ways to use technology safely
KS3 - 7	Evaluate computational abstractions Model state of physical systems Model behaviour of real world problems Understand several key algorithms that reflect computational thinking Use at least one additional programming language (that must be textual) to solve computational problems Make use of appropriate data structures Design modular programs that use procedures or functions Understand uses of Boolean logic in programming Be able to carry out simple operations on binary numbers Understand the software components that make up computer systems Understand how instructions are stored by computer systems Understand how text can be manipulated digitally in the form of binary digits Understand how sounds can be represented digitally in the form of binary digits Understand how pictures can be manipulated digitally in the form of binary digits	Combine multiple applications to achieve challenging goals Analyse data Meet the needs of known users	Revise digital artefacts for a given audience Attend to trustworthiness of digital artefacts Protect online identity Protect privacy
KS3 - 8	Design computational abstractions Model behaviour of physical systems Use logical reasoning to compare the utility of alternative algorithms for the same problem Develop modular programs that use procedures or functions Understand uses of Boolean logic in circuits Understand how computer systems components communicate with one another Understand how computer systems communicate with other systems Understand how instructions are executed by computer systems Understand how sounds can be manipulated digitally in the form of binary digits	Create digital artefacts for a given audience Select multiple applications to achieve challenging goals	Repurpose digital artefacts for a given audience Attend to design of digital artefacts Understand a range of ways to use technology securely Understand a range of ways to use technology responsibly

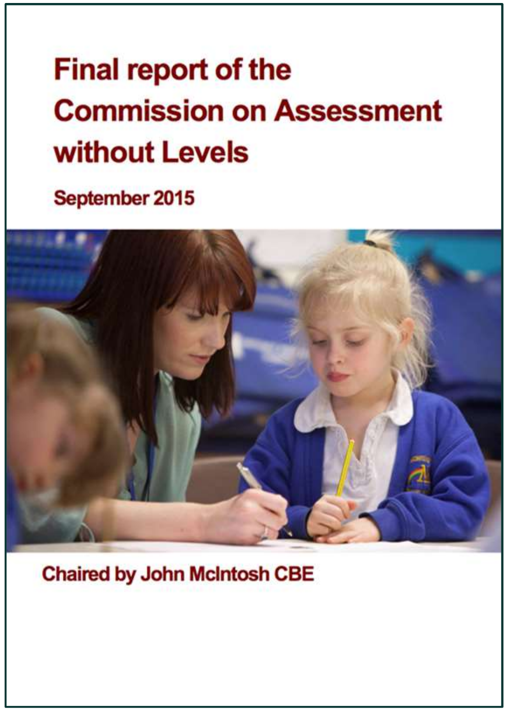
The text above is derived directly from the 2014 national curriculum programmes of study for computing, under the terms of the open government licence 2.0. The organisation in this form is intended to support teachers in forming judgments of their pupils' achievement of and progress towards the statutory attainment targets: "By the end of each key stage, pupils are expected to know, apply and understand the matters, skills and processes specified in the relevant programme of study."

The statements for KS1 and KS2 were included in this form in 'Computing in the national curriculum: a guide for primary teachers' available from Computing at School and Naace. The numbering given here is for convenience only.

Berry and Kemp, 2014

Perguntas de teste

A Comissão recomenda a criação de um banco nacional de perguntas de avaliação a utilizar, tanto para avaliação formativa em sala de aula, para ajudar os professores a avaliar a compreensão de um tópico ou conceito, e para uma avaliação sumária, permitindo aos professores criarem testes personalizados para avaliação no final de um período de tópico ou de ensino.



McIntosh et al 2015

Pedagogia

programação = algoritmos + código

O professor pragmático?

Os instrutores de Ciências da Computação confiam principalmente na intuição e na evidência anedótica para tomar decisões sobre mudanças na sua prática diária de ensino... os instrutores usaram poucos dados empíricos para decidir fazer uma mudança, e para decidir se uma mudança foi bem sucedida ou uma falha.

The Use of Evidence in the Change Making Process of Computer Science Educators

Davide Fossati
Computer Science Department
Carnegie Mellon University
dfossati@cmu.edu

Mark Guzdial
College of Computing
Georgia Institute of Technology
guzdial@cc.gatech.edu

ABSTRACT

This paper explores the issue of what kind of evidence triggers changes in the teaching practice of Computer Science educators, and how educators evaluate the effectiveness of those changes. We interviewed 14 Computer Science instructors from three different institutions. Our study indicates that changes are mostly initiated from instructors' intuition, informal discussion with students, and anecdotal evidence.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer science education

General Terms

Human Factors

1. INTRODUCTION

This paper investigates the question of what kind of evidence triggers and informs changes in the everyday teaching practice of Computer Science educators. As educators, we want to be constantly improving our practice. Success depends on identifying the opportunities or need for improvement, implementing appropriate changes, and then (iteratively) evaluating whether the change met the need.

Our community is productive in providing tools for implementing change. A number of innovative approaches for teaching introductory Computer Science have been designed, such as Beyond LEGOs [3], Media Computation [14], and TeachScheme [9]. Many software systems to support new teaching approaches have been developed, such as Scratch [23] and Alice [6], as well as tools such as algorithm visualization systems [25] and intelligent tutoring systems for Computer Science topics [20, 12].

Researchers are studying the factors that influence adoption of these teaching innovations [21, 22, 16, 8]: What leads a teacher to choose one kind of implementation versus another? These studies explore both catalysts and barriers to change. For example, Ni [22] reports external pressure, limited time, poor background of students, and conflicting views on desired learning outcomes as barriers to change. She also highlights that perceived benefits for students, well-defined pedagogical recommendations, and successful first-hand experience with new approaches are catalysts to innovation adoption. Thus, we know a good bit about innovative approaches, and what leads to their adoption.

In this study, we would like to understand the factors that influence teachers' decisions at a more microscopic level. The teacher has to make decisions about where there is a need for change, and whether the change is effective at meeting the need. For example, why would a teacher change a specific example, a homework assignment, or the format of a group project? Once a change is made, how does a teacher decide whether the change addressed the concern that initiated the change? If those decisions are not made well (e.g., a change is made to something that wasn't really broken, or a change is actually ineffective when judged successful), we are not actually improving practice when we make change.

Specifically, we are interested in understanding the role of evidence in the decision making process of instructors. Researchers have spent some effort in understanding and affirming the important role of evidence in higher-level educational policy and practice [5]. The use of formal evidence is also an essential component of accreditation programs such as ABET [1]. Less attention has been devoted to the use of evidence in the the design decisions in classroom.

Our study aims to provide an initial understanding of this issue in the context of Computer Science education. We interviewed 14 Computer Science instructors from three higher education institutions in the United States, and we extracted the recurring themes in the interviewees' answers. These answers suggest the predominance of instructors' intuition, informal discussion with students, and anecdotal reports as the primary evidence used to inform practical decisions.

2. METHODOLOGY

We interviewed 14 Computer Science instructors, most of them teaching in large research institutions in the Midwest. The background and range of expertise of the interviewees spanned across the majority of sub-disciplines in Computer Science. Some of the instructors were full-time lecturers, whereas others were mostly dedicated to research. The range of classes taught by these instructors also ranged from introductory CS courses for undergraduate students to advanced elective classes taken mostly by graduate students. All the interviews were audio recorded.

SIGCSE'11, March 9–12, 2011, Dallas, Texas, USA.

Fossati and Guzdial, 2011

National Centre for Computing Education

Raspberry Pi

How we teach computing

12 pedagogy principles

Lead with concepts

Support pupils in the acquisition of knowledge, through the use of key concepts, terms, and vocabulary, providing opportunities to build a shared and consistent understanding. Glossaries, concept maps, and displays, along with regular recall and revision, can support this approach.

Unplug, unpack, repack

Teach new concepts by first unpacking complex terms and ideas, exploring these ideas in unplugged and familiar contexts, then repacking this new understanding into the original concept. This approach (semantic waves) can help pupils develop a secure understanding of complex concepts.

Create projects

Use project-based learning activities to provide pupils with the opportunity to apply and consolidate their knowledge and understanding. Design is an important, often overlooked aspect of computing. Pupils can consider how to develop an artefact for a particular user or function, and evaluate it against a set of criteria.

Challenge misconceptions

Use formative questioning to uncover misconceptions and adapt teaching to address them as they occur. Awareness of common misconceptions alongside discussion, concept mapping, peer instruction, or simple quizzes can help identify areas of confusion.

Structure lessons

Use supportive frameworks when planning lessons, such as PRIMM (Predict, Run, Investigate, Modify, Make) and Use-Modify-Create. These frameworks are based on research and ensure that differentiation can be built in at various stages of the lesson.

Work together

Encourage collaboration, specifically using pair programming and peer instruction, and also structured group tasks. Working together stimulates classroom dialogue, articulation of concepts, and development of shared understanding.

Model everything

Model processes or practices — everything from debugging code to binary number conversions — using techniques such as worked examples and live coding. Modelling is particularly beneficial to novices, providing scaffolding that can be gradually taken away.

Add variety

Provide activities with different levels of direction, scaffolding, and support that promote active learning, ranging from highly structured to more exploratory tasks. Adapting your instruction to suit different objectives will help keep all pupils engaged and encourage greater independence.

Make concrete

Bring abstract concepts to life with real-world, contextual examples and a focus on interdependencies with other curriculum subjects. This can be achieved through the use of unplugged activities, proposing analogies, storytelling around concepts, and finding examples of the concepts in pupils' lives.

Read and explore code first

When teaching programming, focus first on code 'reading' activities, before code writing. With both block-based and text-based programming, encourage pupils to review and interpret blocks of code. Research has shown that being able to read, trace, and explain code augments pupils' ability to write code.

Get hands-on

Use physical computing and making activities that offer tactile and sensory experiences to enhance learning. Combining electronics and programming with arts and crafts (especially through exploratory projects) provides pupils with a creative, engaging context to explore and apply computing concepts.

Foster program comprehension

Use a variety of activities to consolidate knowledge and understanding of the function and structure of programs, including debugging, tracing, and Parson's Problems. Regular comprehension activities will help secure understanding and build connections with new knowledge.

Find out more about our principles and add some or all to your personal pedagogy toolkit.

ncce.io/pedagogy

Formação de professores

Questões de ensino

A qualidade de um sistema educativo não pode exceder a qualidade dos seus professores

Interview South Korea 2007, cited in Barber and Mourshed 2007

Computação excecional?

Pedagogia

Competências em TIC

\+ Conhecimento CS = Ensino excecional da computação

Formação, apoio e acreditação

Teach primary computing

Our nationally recognised qualification will support you to demonstrate your commitment to developing your own practice and to computing as a school subject.

Find out more

Browse primary certificate courses

Teacher CPD

Databases

When:
Series starts Tue, 20 Apr 2021
16:00 — 18:00

Location: Online

View details

Teacher CPD

Data Structures

When:
Series starts Thu, 22 Apr 2021
9:00 — 12:00

Location: Online

View details

Teacher CPD

Web technologies

When:
Series starts Wed, 5 May 2021
9:30 — 11:30

Location: Online

View details

GCSE computer science subject knowledge

Computer Science Accelerator is a flexible professional development programme designed to equip you with the subject knowledge to teach GCSE computer science.

Find out more

Browse GCSE certificate courses

Teach secondary computing

Following on from the Computer Science Accelerator, this programme will give you confidence to take your teaching to the next level and to apply those skills in the classroom.

Find out more

Browse Teach secondary courses

29,500

teachers engaged, representing **8,500** primary schools and **3,000** secondary schools.

7,600

teachers have benefitted from NCCE continuing professional development (CPD).

Teach Computing Curriculum launched, including

500

hours of learning materials from key stages **1** to **4**.

1,300

Computer Science Accelerator graduates trained to teach GCSE Computer Science.

125,000

units of work downloaded from the Teach Computing Curriculum since September 2020.

34

Computing Hubs acting as local champions for Computer Science and delivering face-to-face, in-school and remote support.

1 million

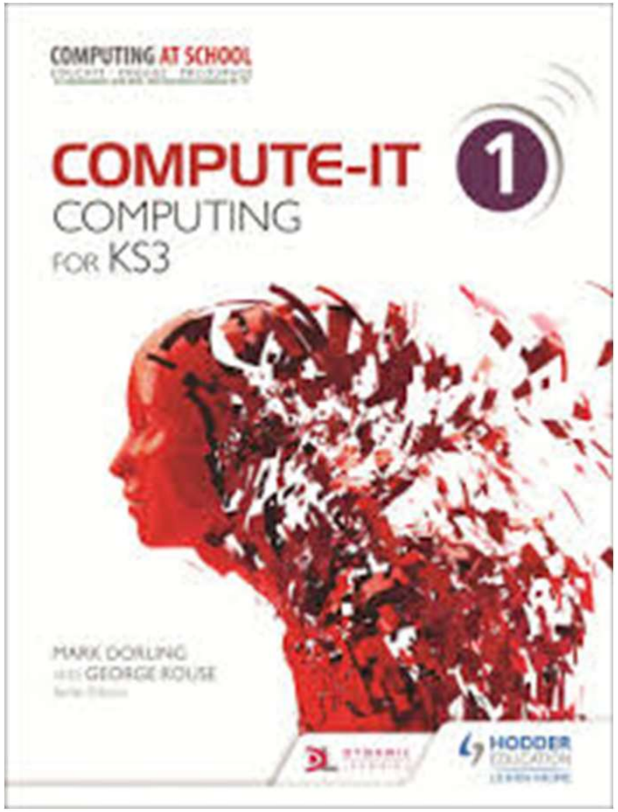
questions answered through Isaac Computer Science, supporting A level Computer Science.

275

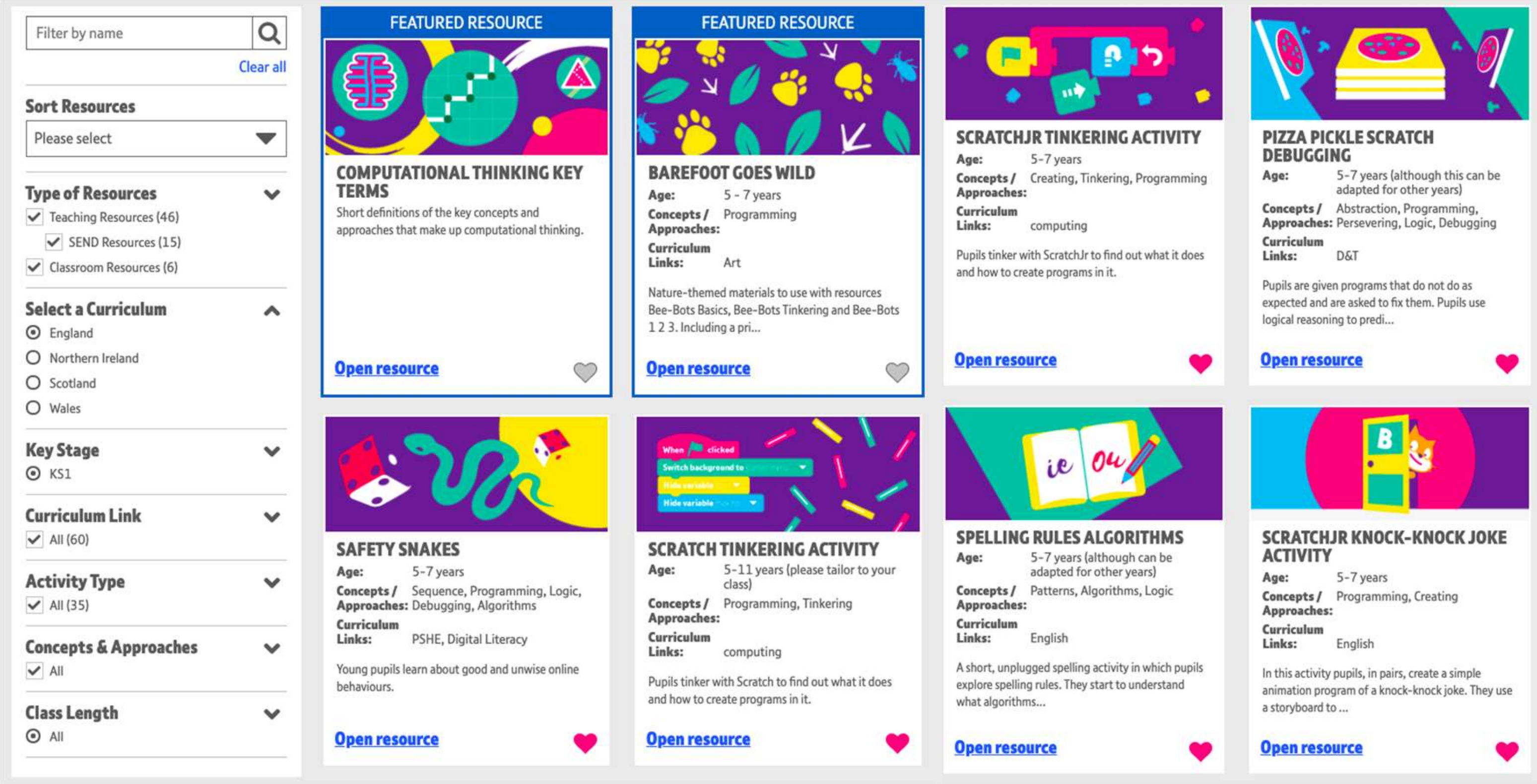
Computing at School (CAS) Communities of Practice providing peer support and networking.

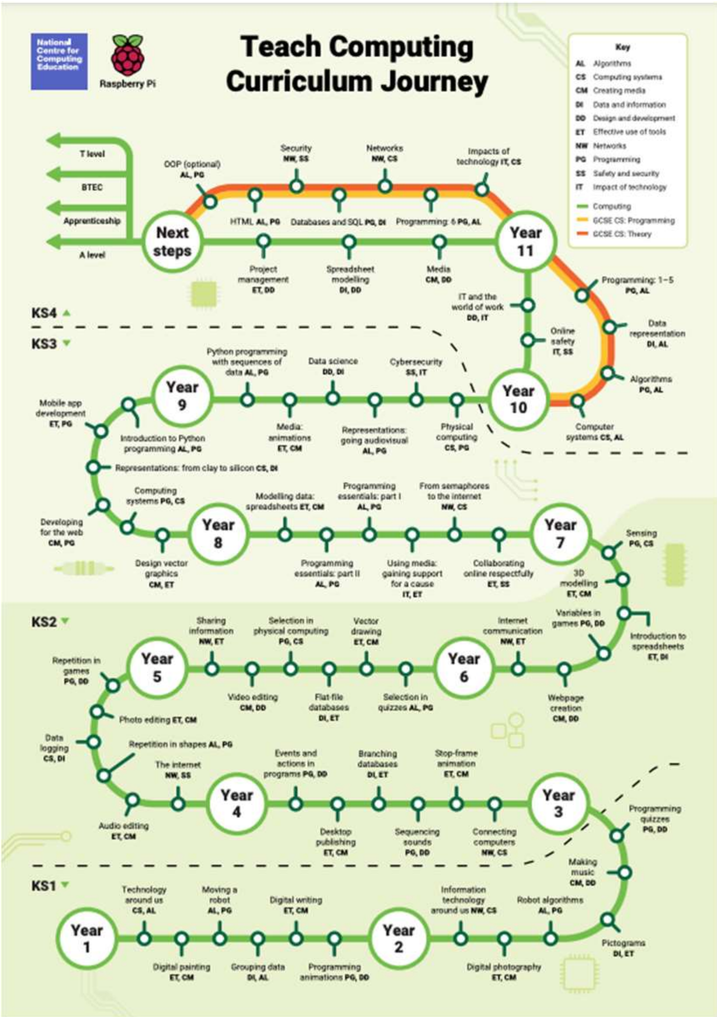
Materiais pedagógicos

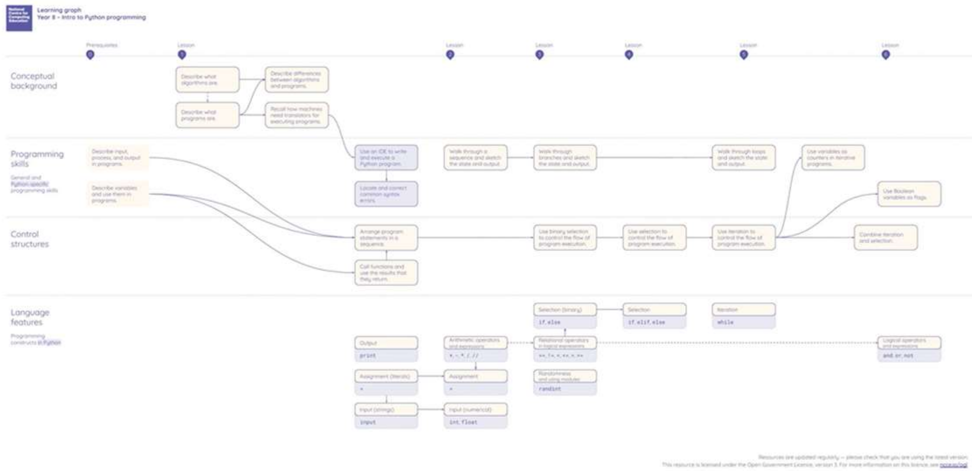

SWITCHED ON
Computing
YEAR 1
2
3
4
5
6


COMPUTING AT SCHOOL
COMPUTE-IT 1
COMPUTING
FOR KS3
MARK DORLING
GEORGE ROUSE
HODDER EDUCATION

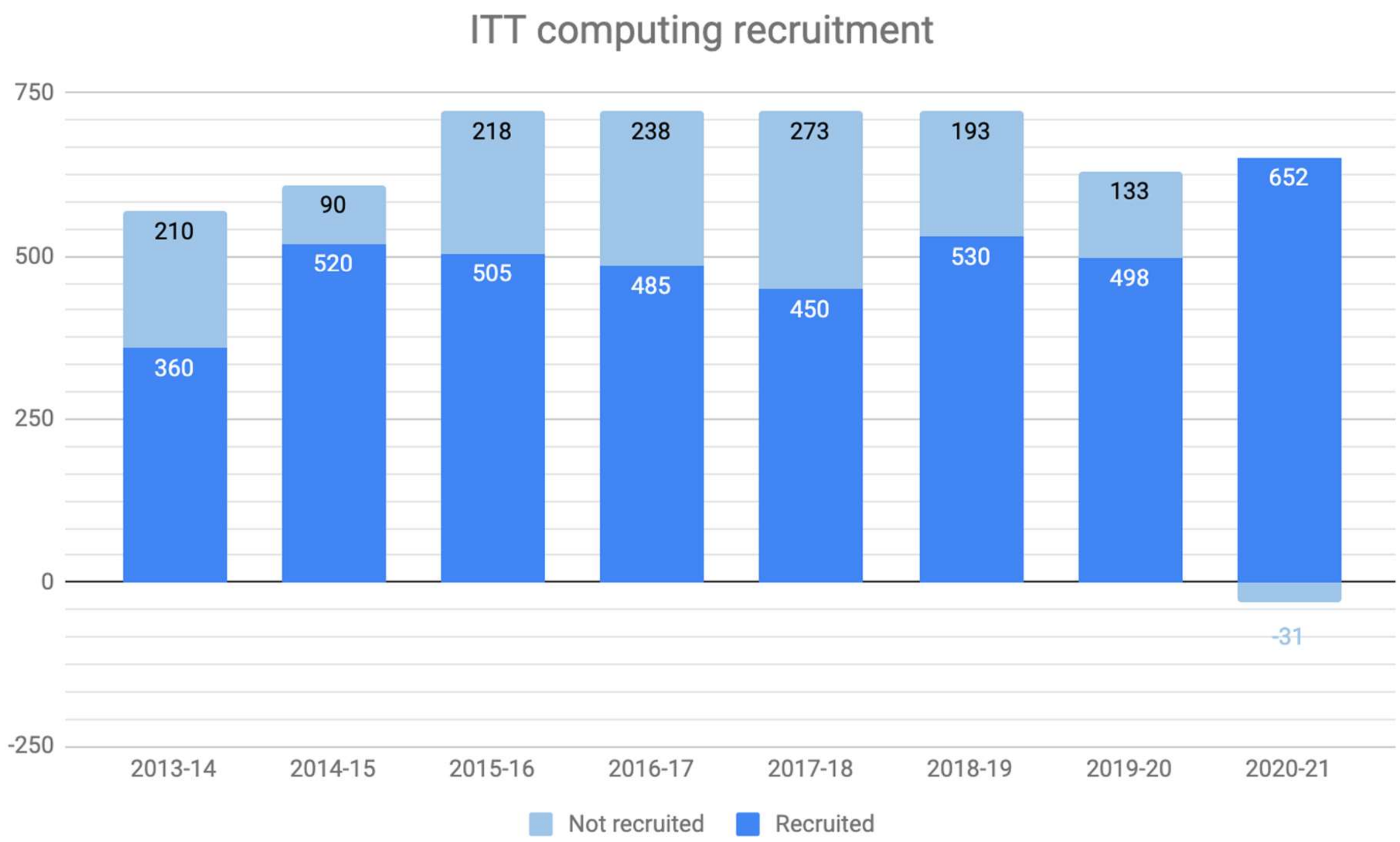
Computação descalça




National Centre for Computing Education
Raspberry Pi
Teach Computing Curriculum Journey
Key
AL Algorithms
CS Computing systems
CM Creating media
DI Data and information
DD Design and development
ET Effective use of tools
NW Networks
PG Programming
SS Safety and security
IT Impact of technology
Computing
GCSE CS: Programming
GCSE CS: Theory
T level
BTEC
Apprenticeship
A level
Next steps
OOP (optional) AL, PG
Security NW, SS
Networks NW, CS
Impacts of technology IT, CS
HTML AL, PG
Databases and SQL PG, DI
Programming: 6 PG, AL
Year 11
Project management ET, DD
Spreadsheet modelling DI, DD
Media CM, DD
Programming: 1–5 PG, AL
IT and the world of work DD, IT
Data representation DI, AL
Online safety IT, SS
Algorithms PG, AL
Year 10
Computer systems CS, AL
KS4
KS3
Python programming with sequences of data AL, PG
Data science DD, DI
Cybersecurity SS, IT
Year 9
Mobile app development ET, PG
Introduction to Python programming AL, PG
Media: animations ET, CM
Representations: going audiovisual AL, PG
Physical computing CS, PG
Representations: from clay to silicon CS, DI
Computing systems PG, CS
Modelling data: spreadsheets ET, CM
Programming essentials: part I AL, PG
From semaphores to the internet NW, CS
Developing for the web CM, PG
Year 8
Year 7
Sensing PG, CS
Design vector graphics CM, ET
Programming essentials: part II AL, PG
Using media: gaining support for a cause IT, ET
Collaborating online respectfully ET, SS
3D modelling ET, CM
KS2
Sharing information NW, ET
Selection in physical computing PG, CS
Vector drawing ET, CM
Internet communication NW, ET
Variables in games PG, DD
Introduction to spreadsheets ET, DI
Repetition in games PG, DD
Year 5
Year 6
Video editing CM, DD
Flat-file databases DI, ET
Selection in quizzes AL, PG
Webpage creation CM, DD
Photo editing ET, CM
Data logging CS, DI
Repetition in shapes AL, PG
The internet NW, SS
Events and actions in programs PG, DD
Branching databases DI, ET
Stop-frame animation ET, CM
Year 4
Year 3
Programming quizzes PG, DD
Audio editing ET, CM
Desktop publishing ET, CM
Sequencing sounds PG, DD
Connecting computers NW, CS
Making music CM, DD
KS1
Technology around us CS, AL
Moving a robot AL, PG
Digital writing ET, CM
Information technology around us NW, CS
Robot algorithms AL, PG
Year 1
Year 2
Digital painting ET, CM
Grouping data DI, AL
Programming animations PG, DD
Digital photography ET, CM
Pictograms DI, ET


Conceptual background
Programming skills
Control structures
Language features

Incentivos e motoristas


ITT computing recruitment
750
500
250
0
-250
210
360
90
520
218
505
238
485
273
450
193
530
133
498
652
-31
2013-14
2014-15
2015-16
2016-17
2017-18
2018-19
2019-20
2020-21
Not recruited
Recruited

DfE, 2020

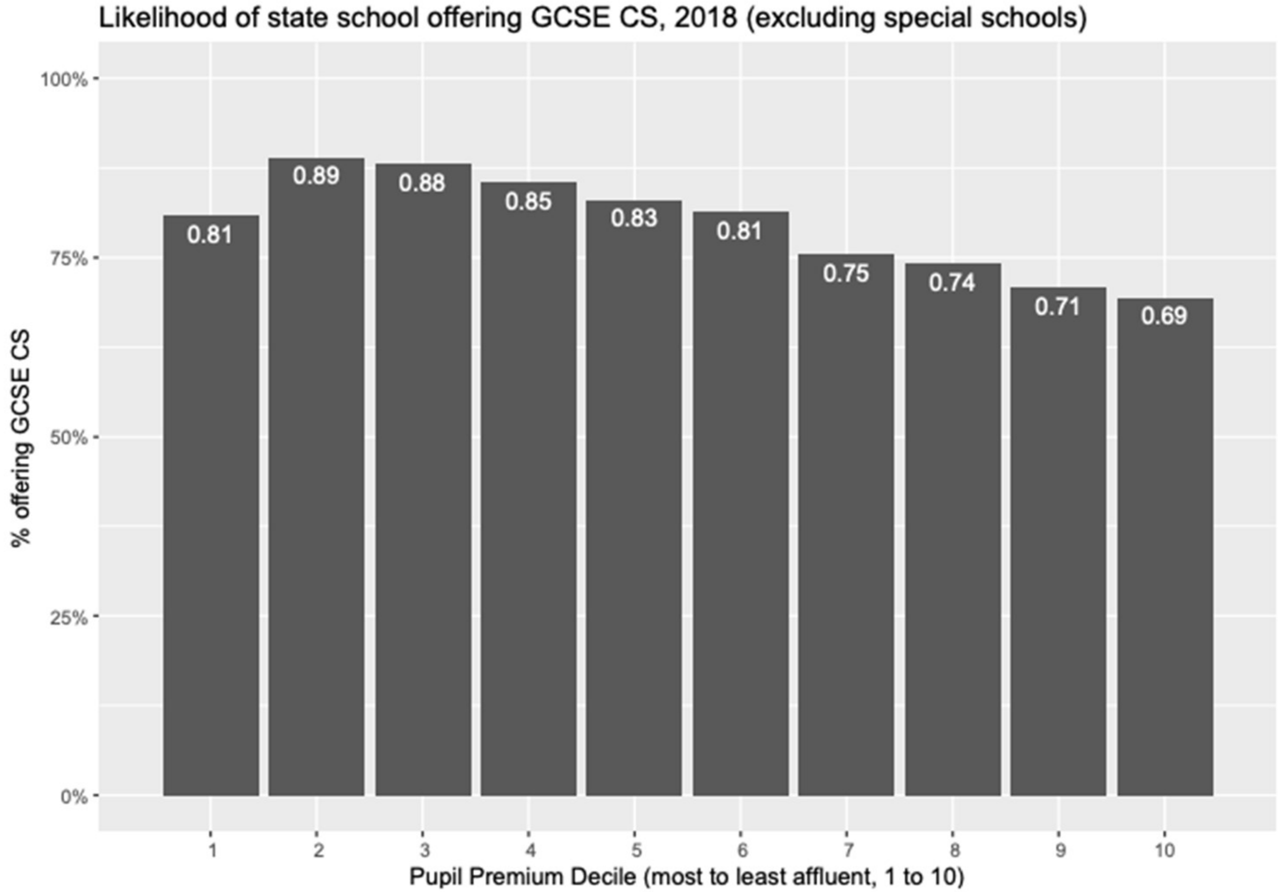
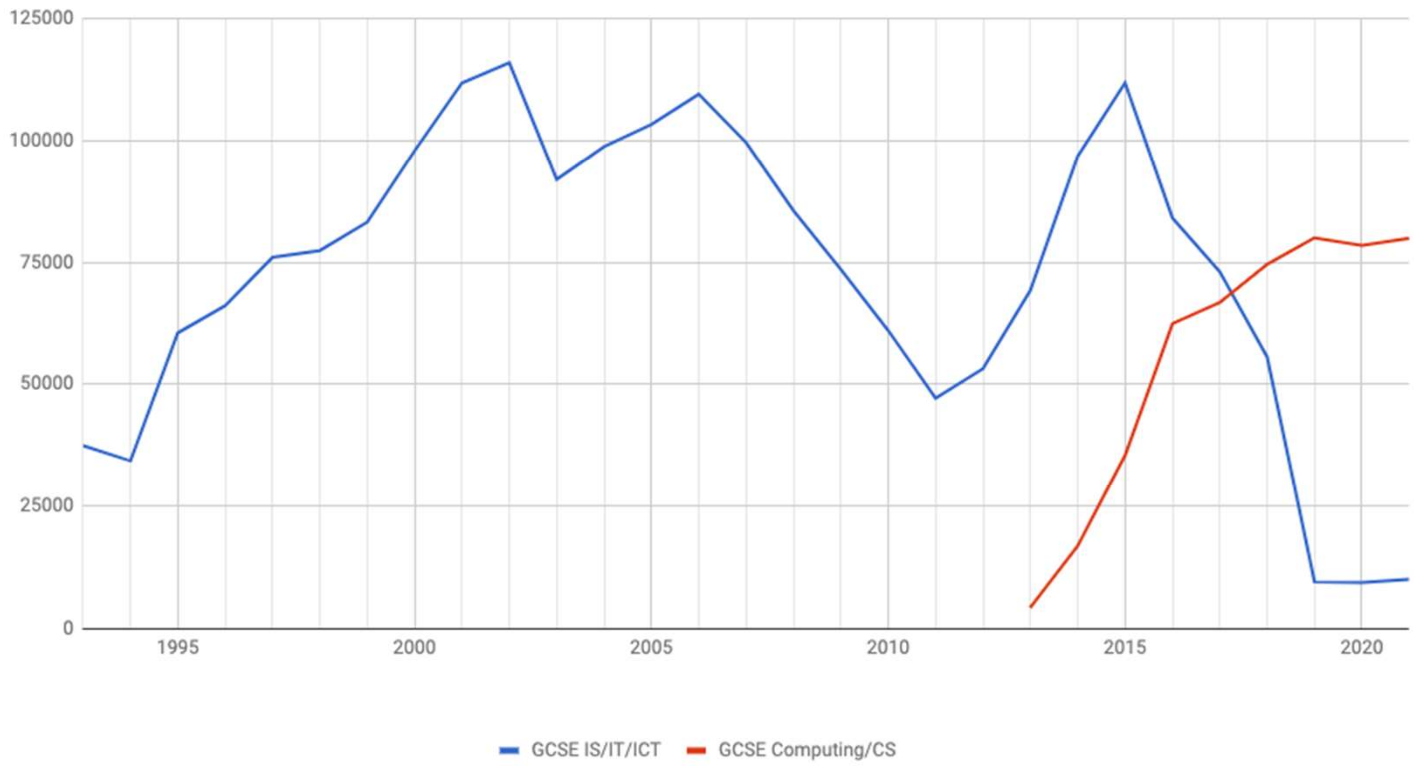
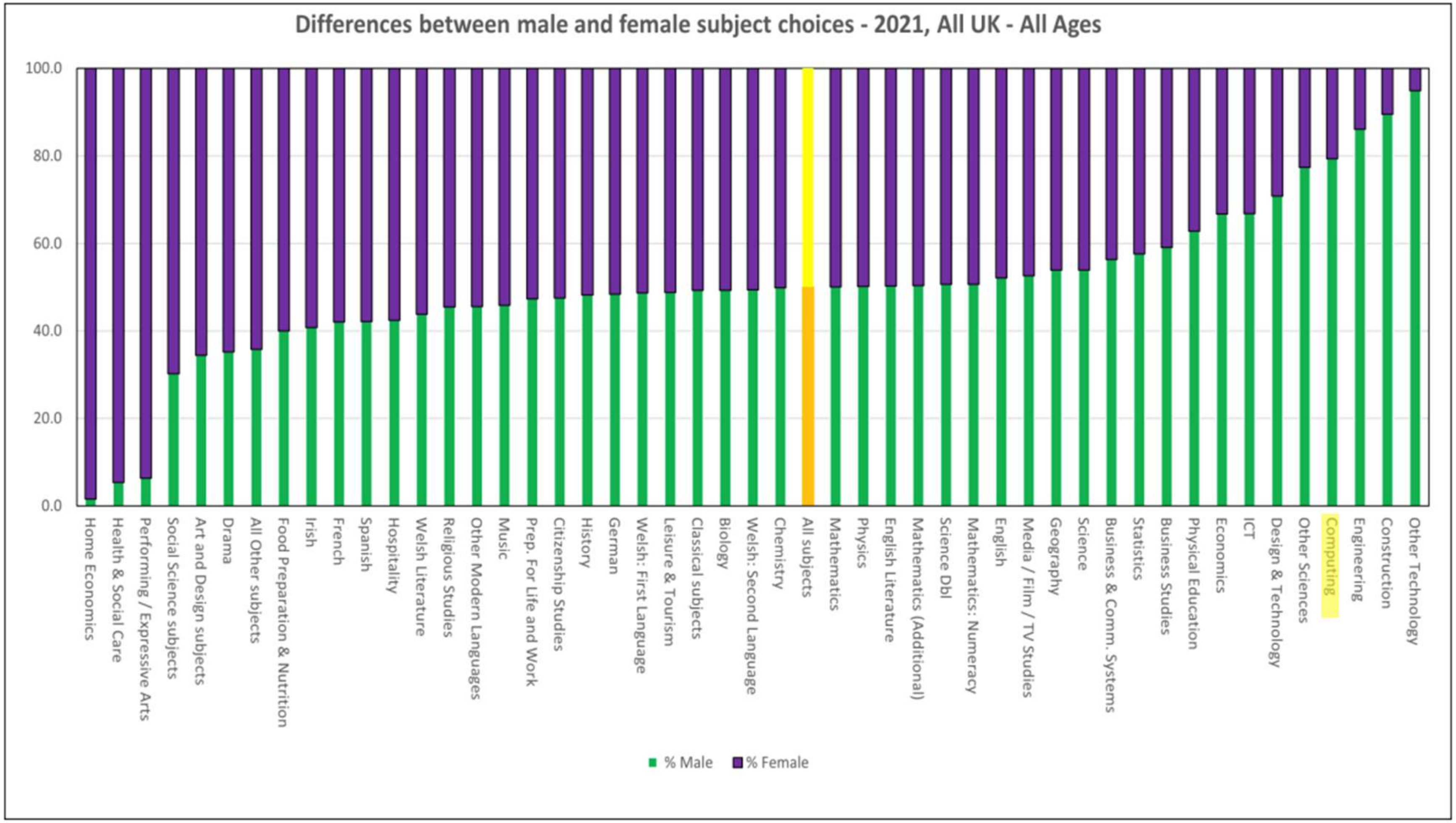


Figure 1: Likelihood of state funded schools offering GCSE CS, 2018, by IDACI decile (excluding special schools)

Kemp and Berry, 2019

GCSE IS/IT/ICT and GCSE Computing/CS





Differences between male and female subject choices - 2021, All UK - All Ages
100.0
80.0
60.0
40.0
20.0
0.0
Home Economics
Health & Social Care
Performing / Expressive Arts
Social Science subjects
Art and Design subjects
Drama
All Other subjects
Food Preparation & Nutrition
Irish
French
Spanish
Hospitality
Welsh Literature
Religious Studies
Other Modern Languages
Music
Prep. For Life and Work
Citizenship Studies
History
German
Welsh: First Language
Leisure & Tourism
Classical subjects
Biology
Welsh: Second Language
Chemistry
All subjects
Mathematics
Physics
English Literature
Mathematics (Additional)
Science Dbl
Mathematics: Numeracy
English
Media / Film / TV Studies
Geography
Science
Business & Comm. Systems
Statistics
Business Studies
Physical Education
Economics
ICT
Design & Technology
Other Sciences
Computing
Engineering
Construction
Other Technology
% Male
% Female

Observações finais

Escrever uma estrutura curricular não é suficiente
Plano de coerência
A formação e os recursos são vitais para o sucesso
A pedagogia?
Tente fazer a transição certa
CS para todos

Perguntas?

m.berry@roehampton.ac.uk
@mberry
milesberry.net