University of Roehampton London

Computing in English schools: this is for everyone

Miles Berry
@mberry
These slides: bit.ly/erte15

17 October 2015

Outline

Rationales

From ICT to Computing

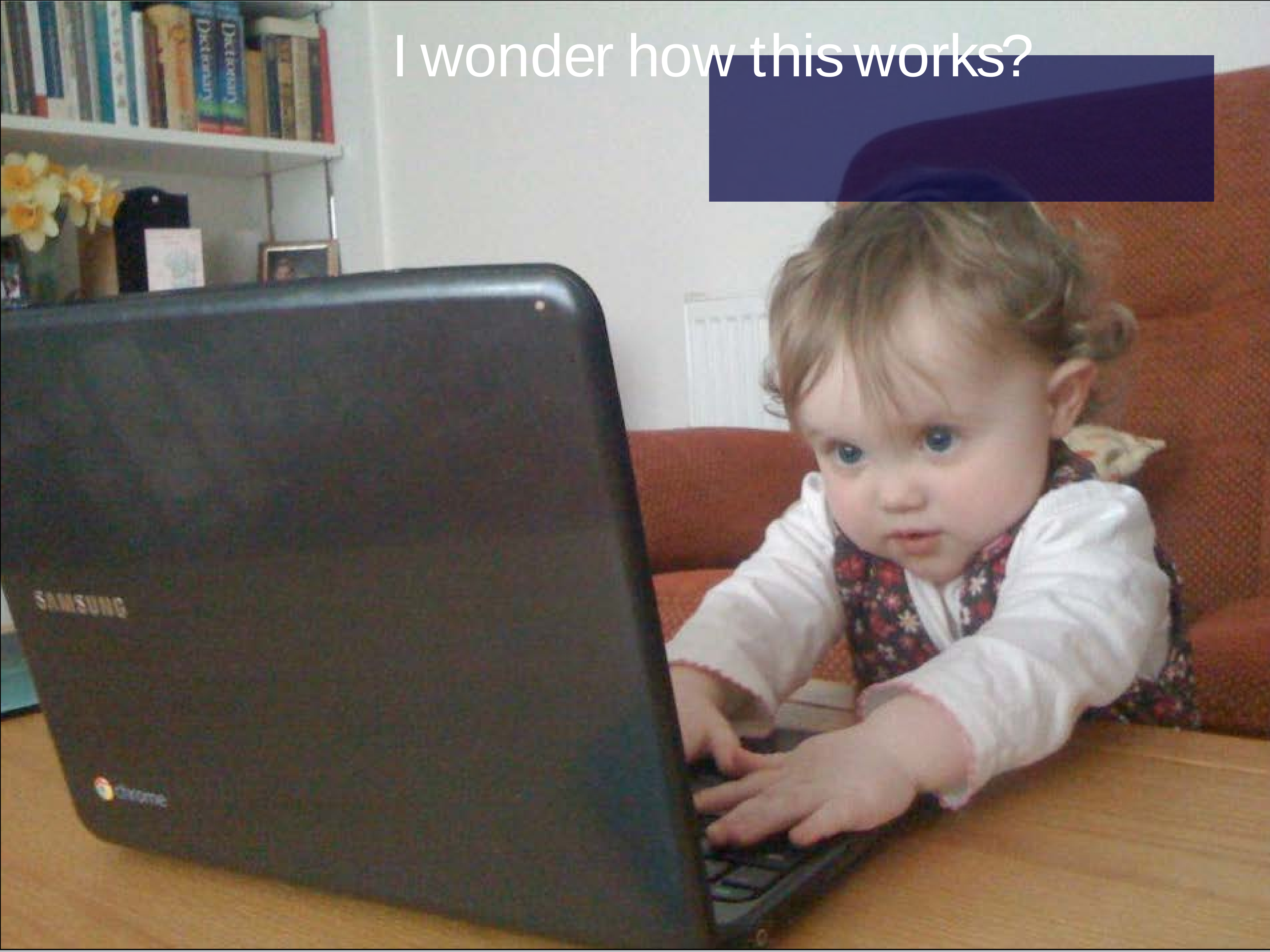
Curriculum content

Support

Challenges ahead

Why teach CS to school children?

I wonder how this works?

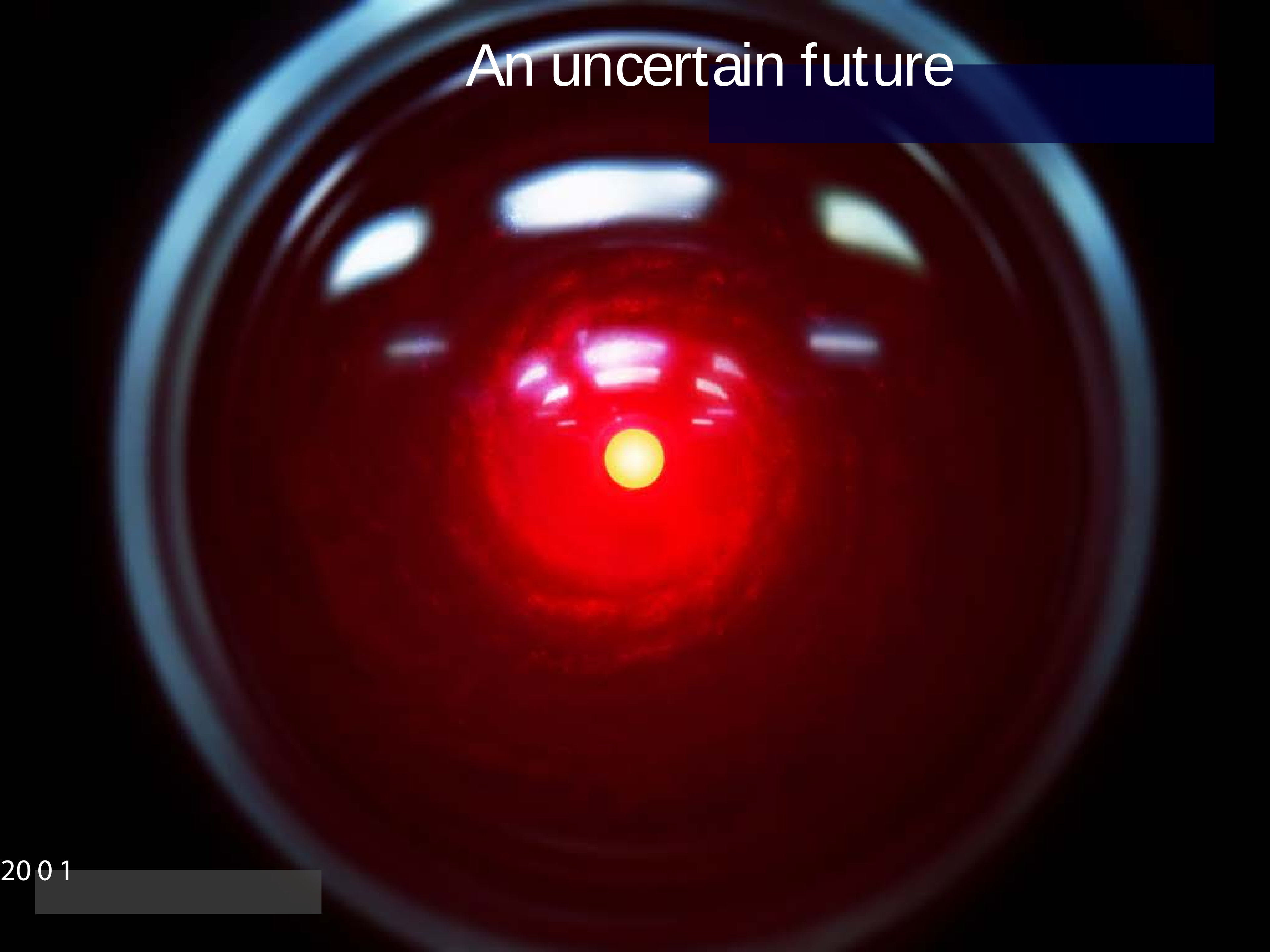


The zeroth science

machine.com

UK plc / Portugal Lda

An uncertain future



A liberal education in the third millennium

Tools to think with

Mindstorms

Designing a computing curriculum

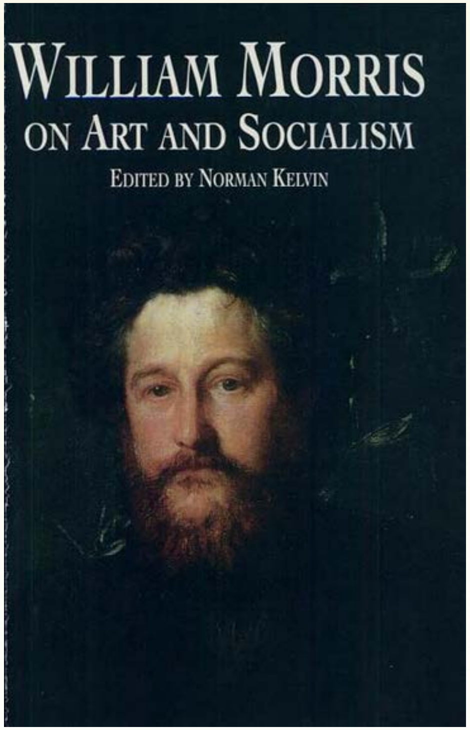
From consumers to creators

Users	Makers
Communicators	Collaborators
Digitally literate	Digitally critical
Safe	Responsible
Skills	Understanding
Magic	Knowledge

Beauty or utility?

If you want a golden rule that will fit everybody, this is it:

Have nothing in your houses that you do not know to be useful, or believe to be beautiful.



Morris, 1880

Computer Science
Information Technology
Digital Literacy

Foundations

Applications

Implications

Aims

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

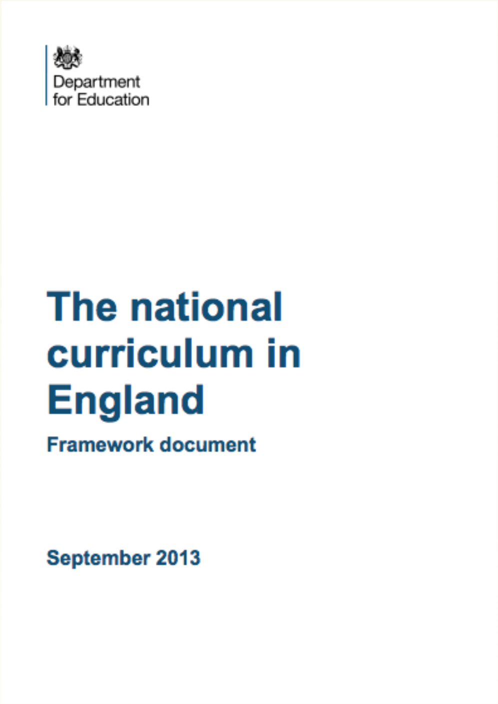
can evaluate and apply **information technology**, including new or unfamiliar technologies, analytically to solve problems

are responsible, competent, confident and creative **users** of information and communication technology

DfE, 2013

Computing

A high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world.


Department for Education

The national curriculum in England

Framework document

September 2013

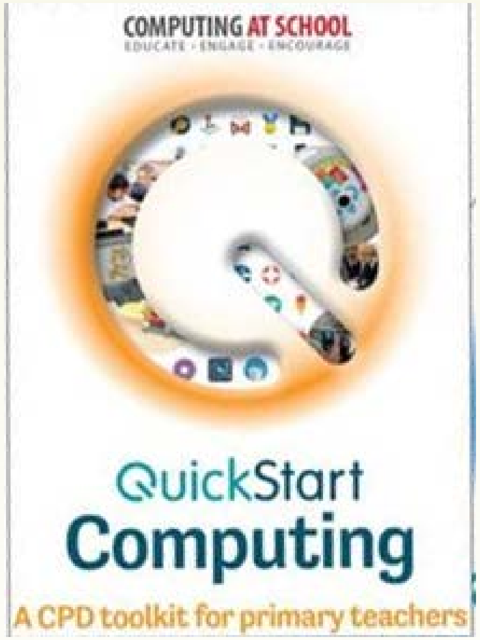
DfE, 2013

Computational thinking

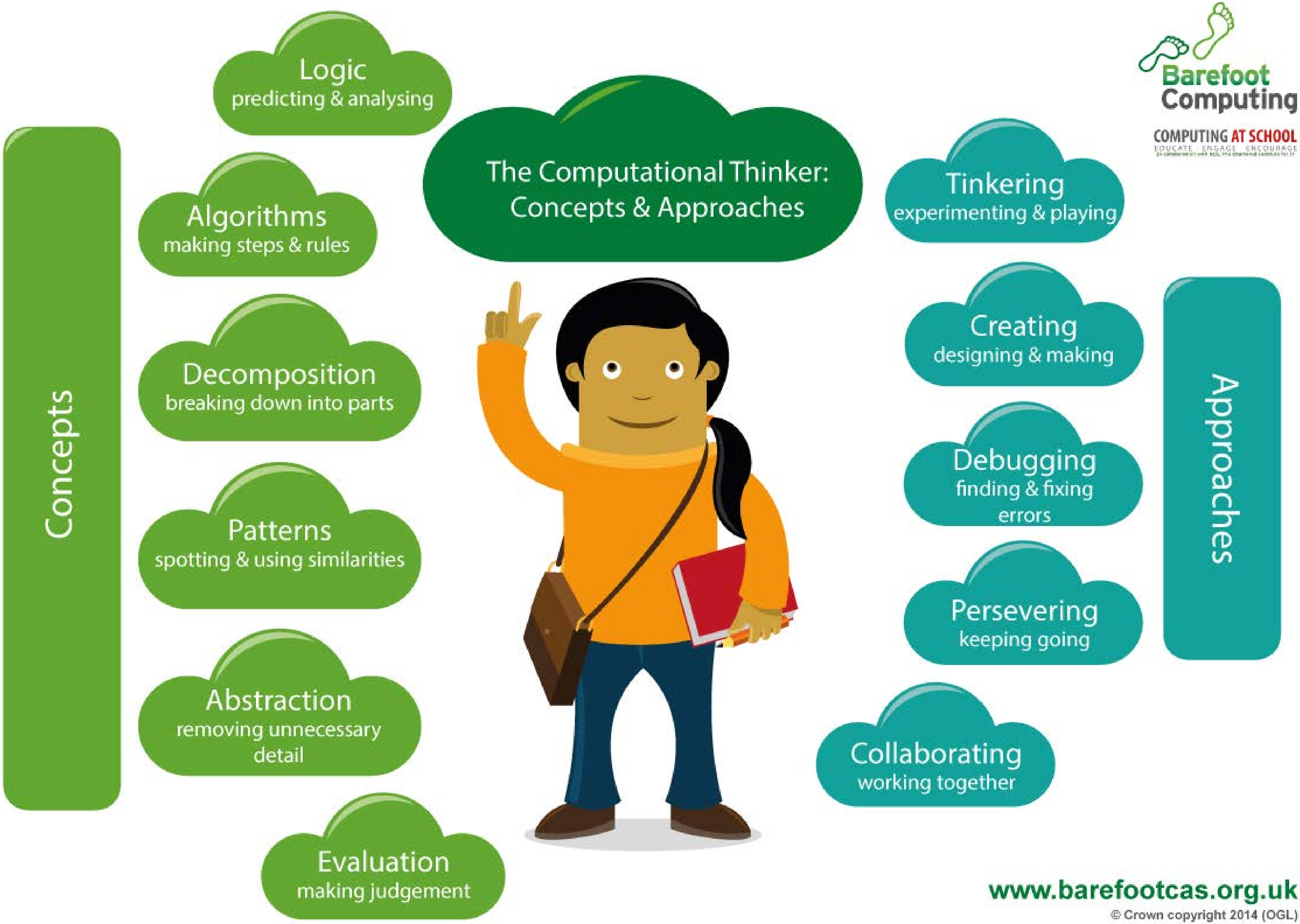
Getting computers to help us to solve problems is a two-step process:

1. think about the steps to solve a problem or the rules that govern the system
2. use your technical skills to get the computer working on the problem.

Computational thinking is the first of these. It describes **the concepts, processes and approaches we draw on when thinking about problems or systems in such a way that a computer can help us with these.**



Berry 2015



Barefoot would like to acknowledge the work of Julia Briggs and the eLIM team at Somerset County Council for their contribution to this poster.

EYFS

Before 5

Before 5

	A Unique Child: observing how a child is learning
Creating and Thinking Critically *thinking*	**Having their own ideas** • Thinking of ideas • Finding ways to solve problems • Finding new ways to do things
	Making links • Making links and noticing patterns in their experience • Making predictions • Testing their ideas • Developing ideas of grouping, sequences, cause and effect
	Choosing ways to do things • Planning, making decisions about how to approach a task, solve a problem and reach a goal • Checking how well their activities are going • Changing strategy as needed • Reviewing how well the approach worked

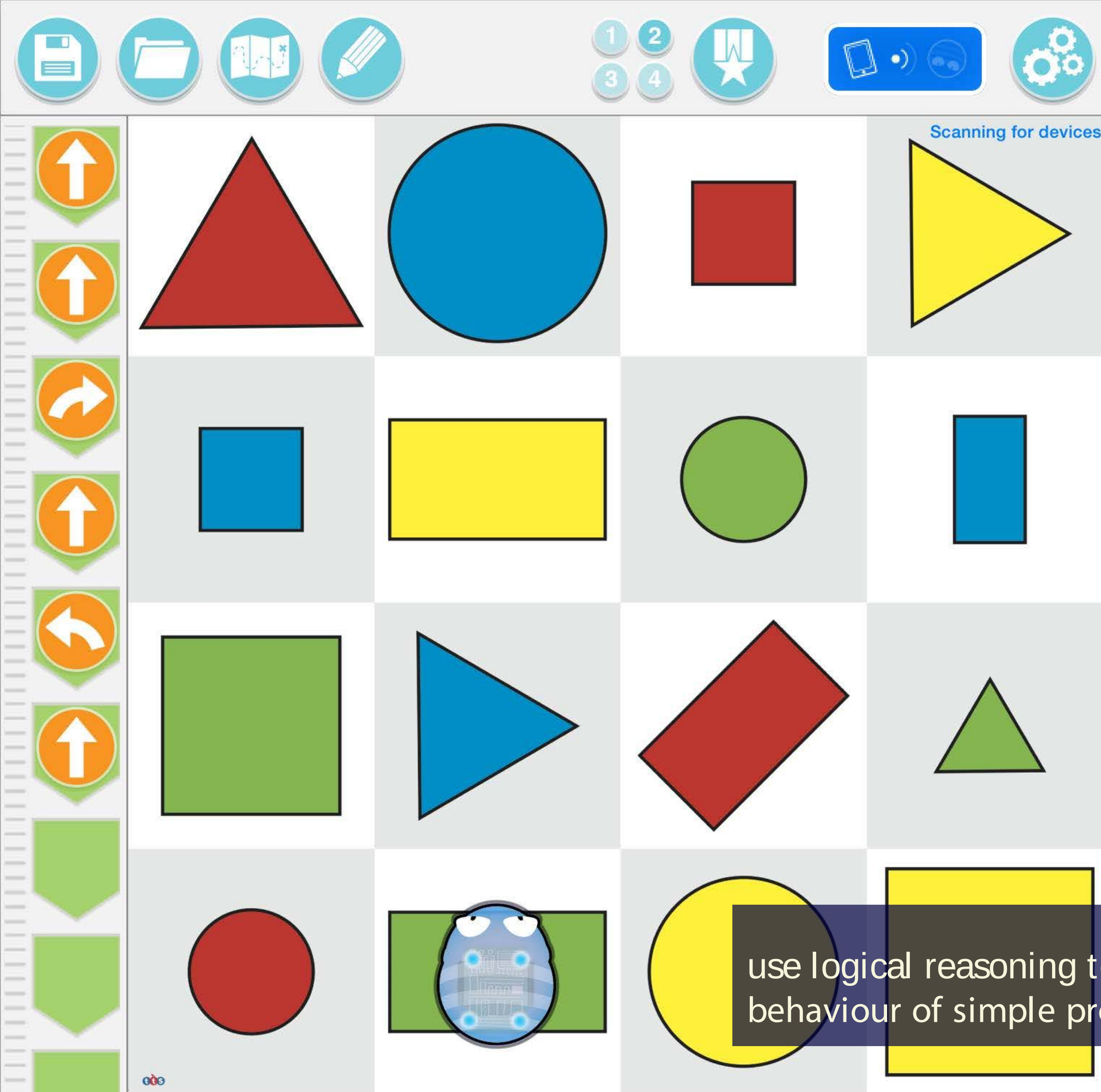

Thinkosaur
Tryosaur
Explorosaur

Key Stage 1

5 - 7 years old

understand what algorithms are, how they are implemented as programs on digital devices, and that programs execute by following a sequence of instructions

create and debug simple programs

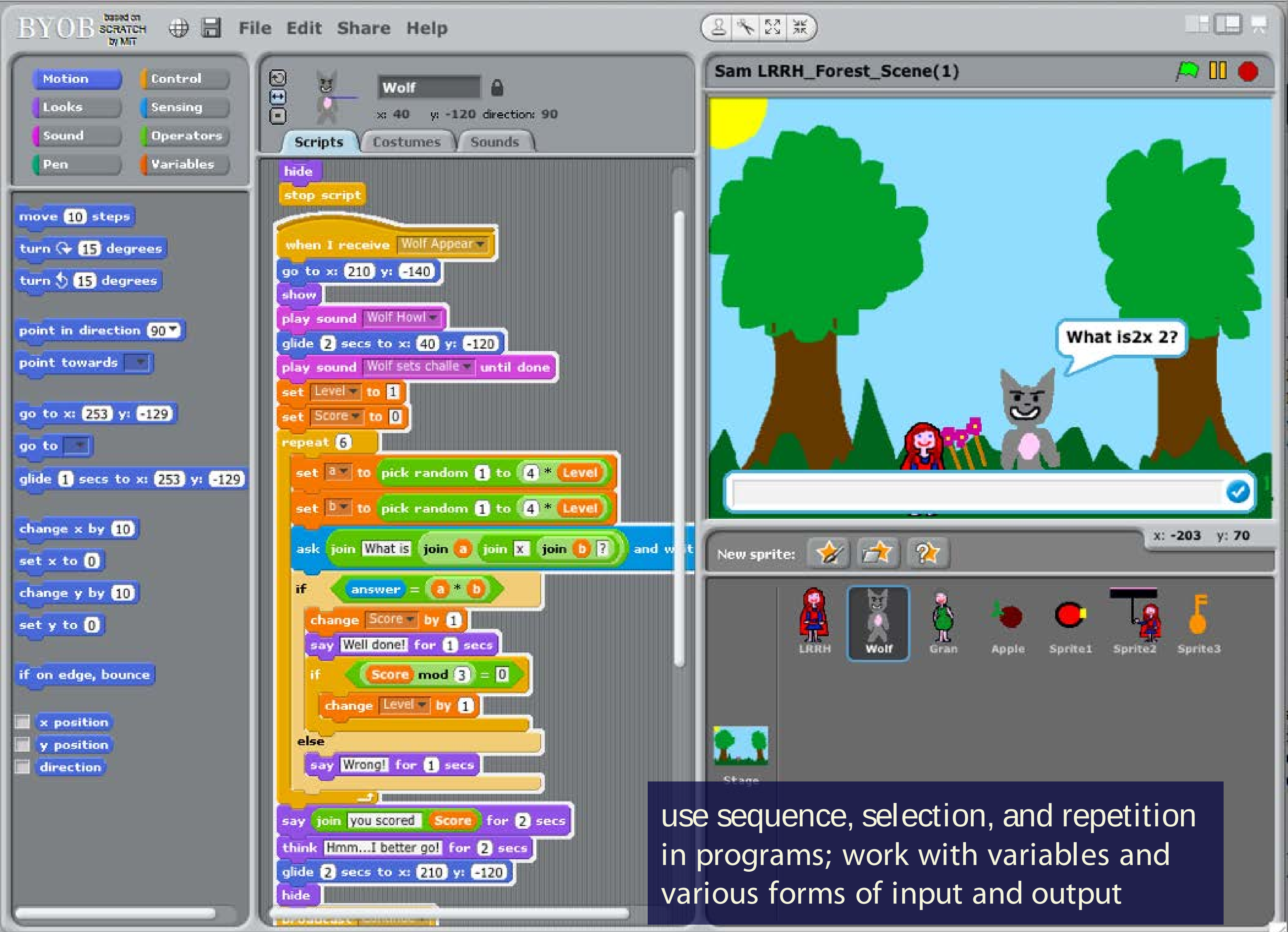


use logical reasoning to predict the behaviour of simple programs

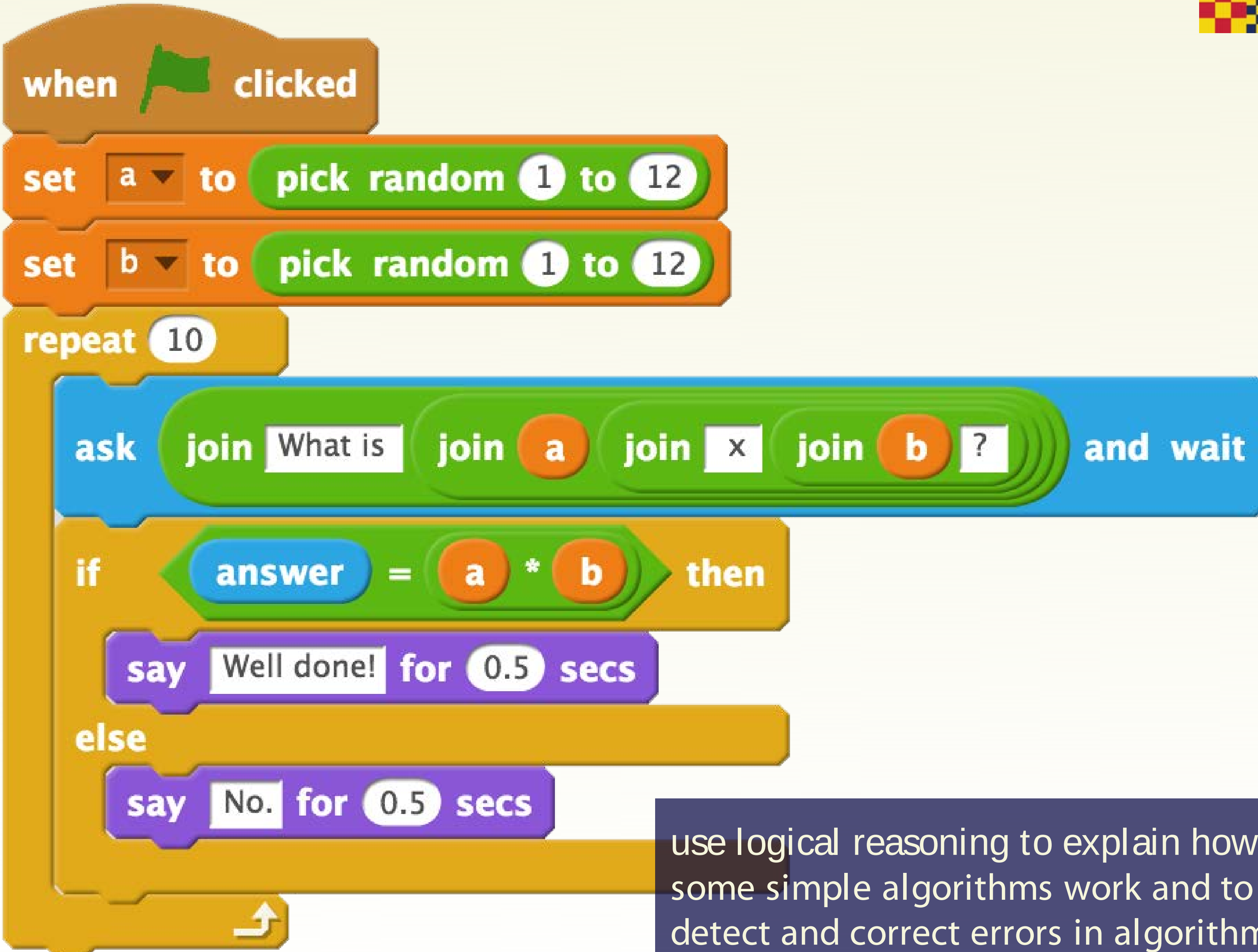
recognise common uses of information technology beyond school

Key Stage 2

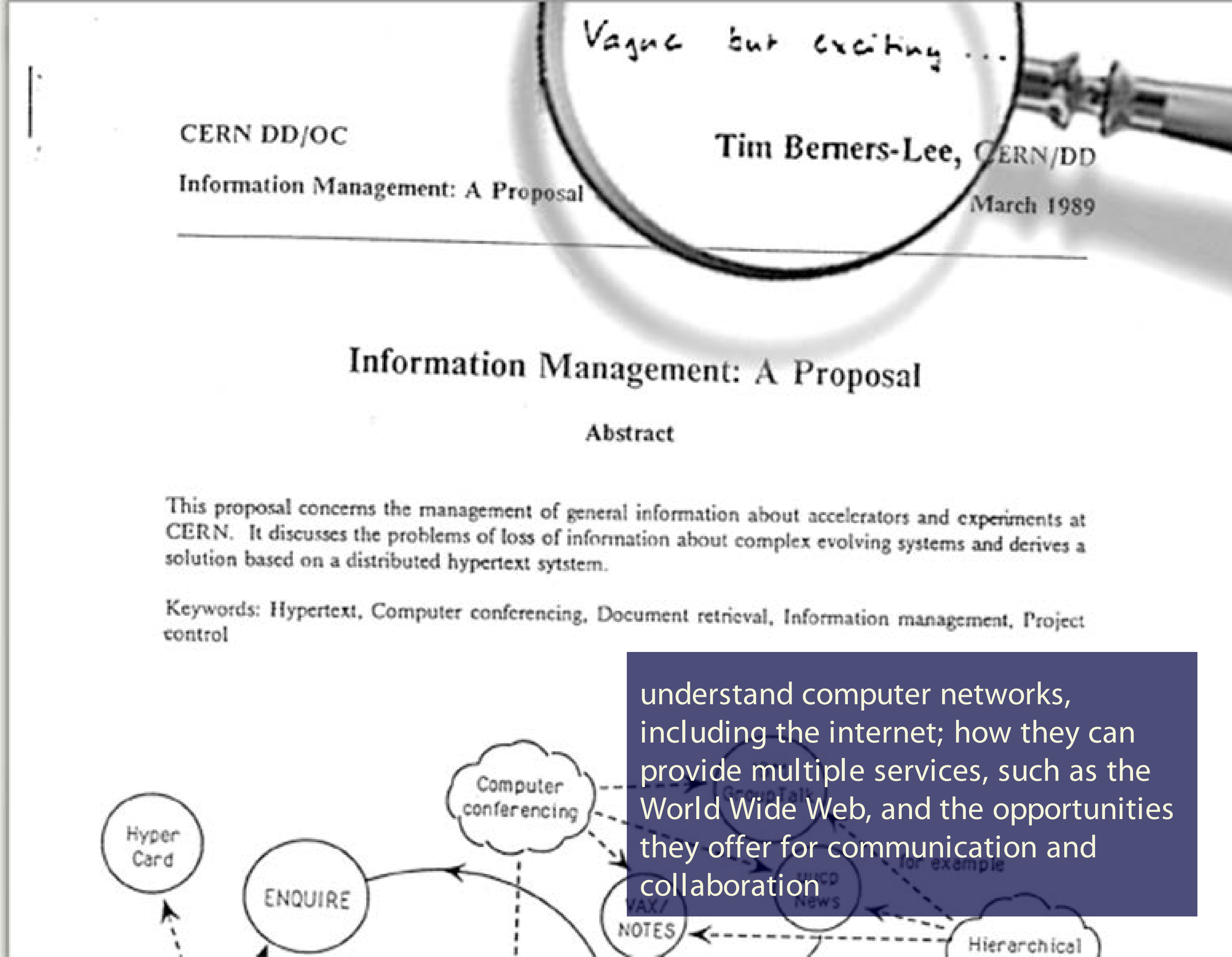
7 - 11 years old



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



CERN DD/OC

Information Management: A Proposal

Tim Berners-Lee, CERN/DD

March 1989

Information Management: A Proposal

Abstract

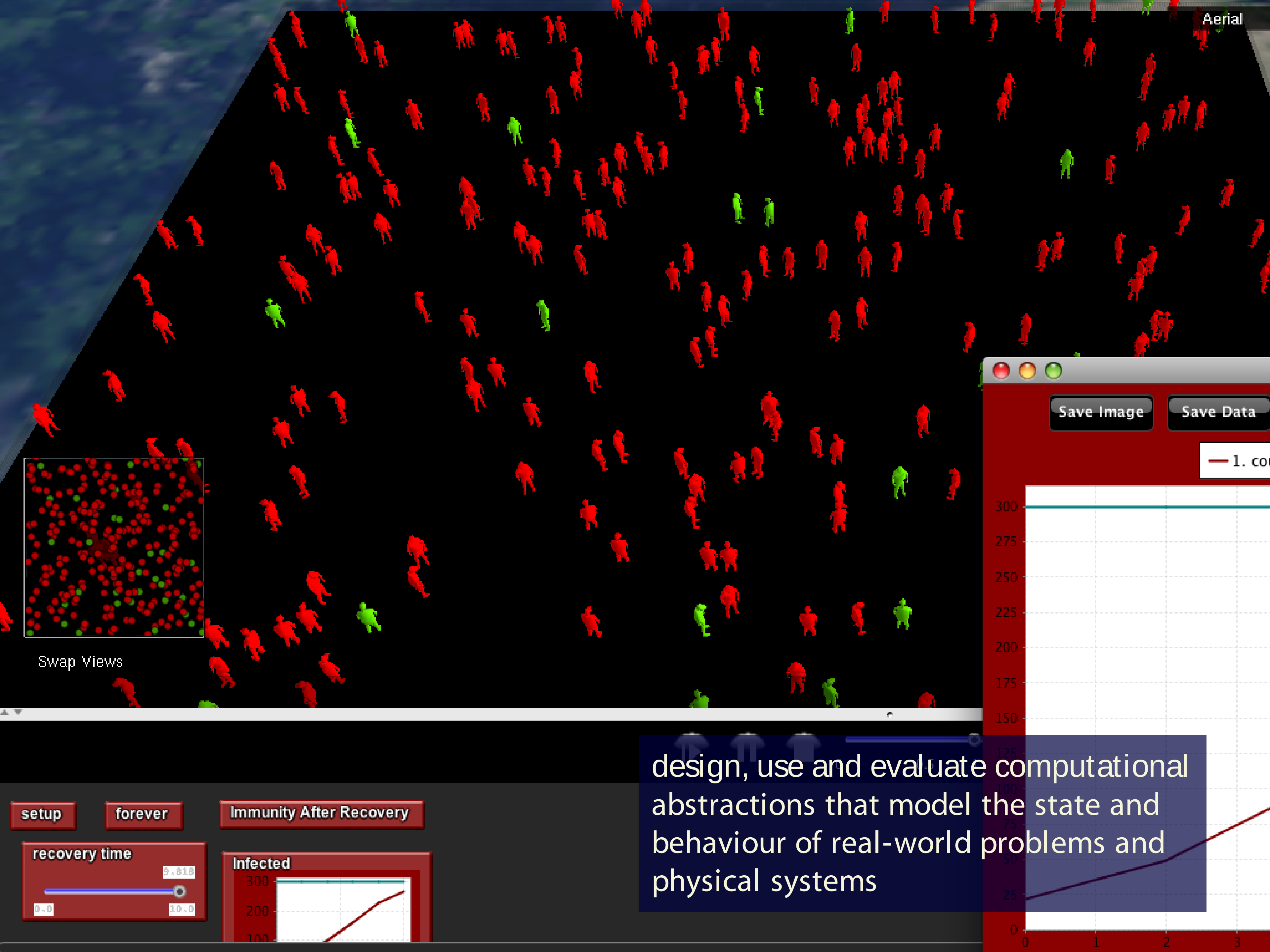
This proposal concerns the management of general information about accelerators and experiments at CERN. It discusses the problems of loss of information about complex evolving systems and derives a solution based on a distributed hypertext sytstem.

Keywords: Hypertext, Computer conferencing, Document retrieval, Information management, Project control

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

Key Stage 3

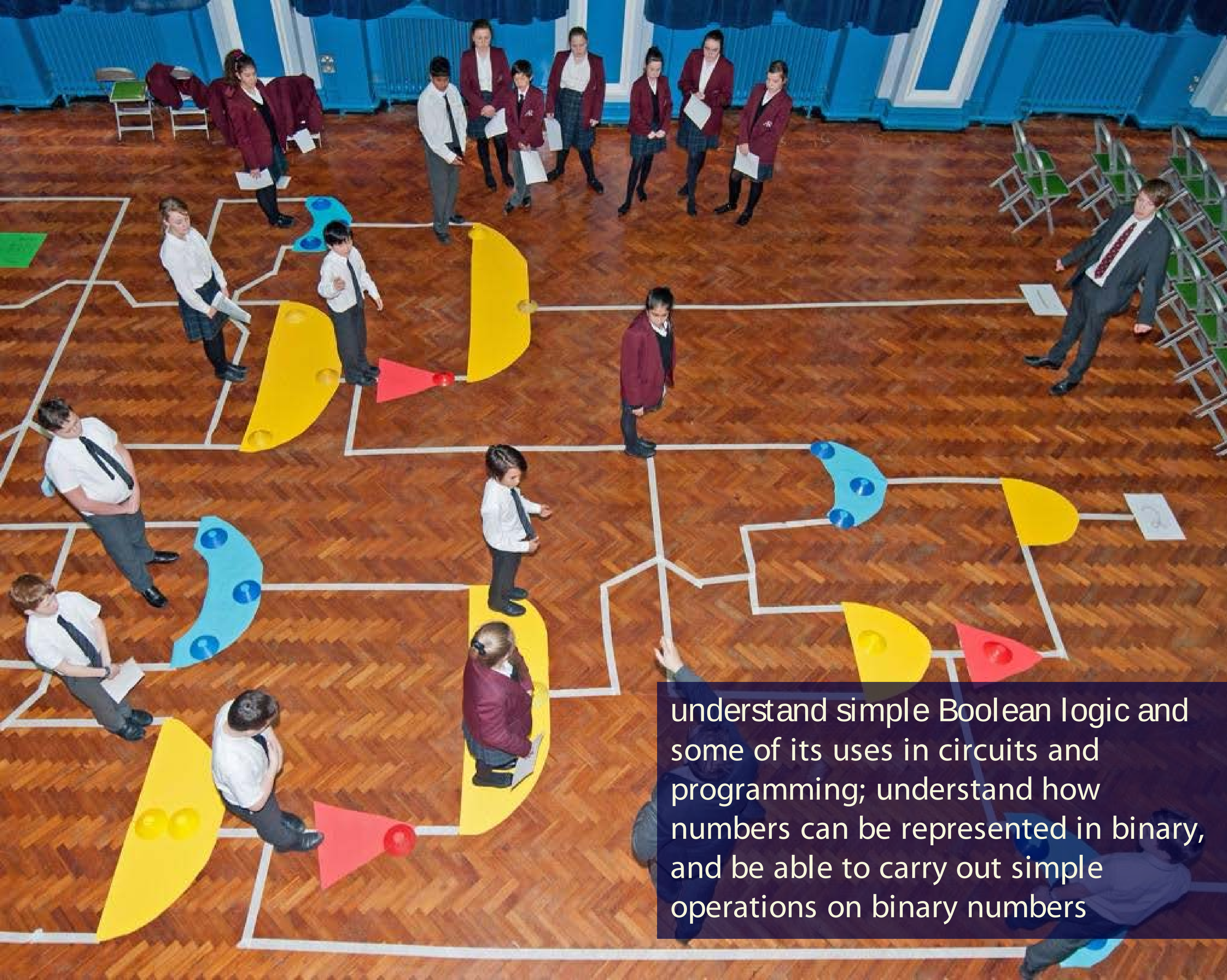
11 - 14 years old



design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems

```
import random
for i in range(5):
      a = random.randint(1,12)
      b = random.randint(1,12)
      question = "What is "+str(a)+" x "+str(b)+"?"
      answer = int(input(question))
      if answer == a*b:
      print("Well done!")
      else:
      print("No.")
```

use 2 or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures; design and develop modular programs that use procedures or functions



understand simple Boolean logic 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

Key Stage 4

14 - 16 years old

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 report a range of concerns

The content for computer science GCSEs

Introduction

1. The GCSE subject content sets out the knowledge, understanding and skills common to all GCSE specifications in a given subject. Together with the assessment objectives it provides the framework within which the awarding organisations create the detail of their specifications, so ensuring progression from key stage 3 national curriculum requirements and the possibilities for development into A level.

Subject aims and learning outcomes

2. All specifications in computer science must build on the knowledge, understanding and skills established through the computer science elements of the programme of study for computing at key stage 3, satisfy the computer science elements of computing at key stage 4 and enable students to progress into further learning and/or employment.

3. GCSE specifications in computer science should enable students to:

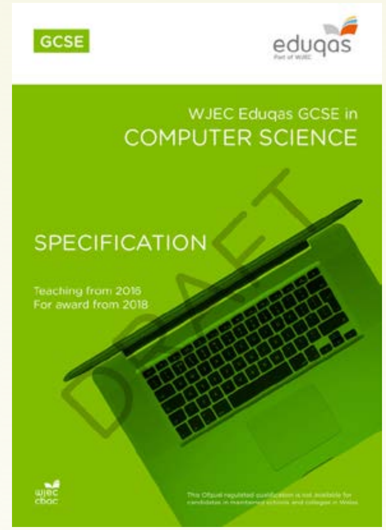
- understand and apply the fundamental principles and concepts of computer science, including abstraction, decomposition, logic, algorithms, and data representation
- analyse problems in computational terms through practical experience of solving such problems, including designing, writing and debugging programs
- think creatively, innovatively, analytically, logically and critically
- understand the components that make up digital systems, and how they communicate with one another and with other systems
- understand the impacts of digital technology to the individual and to wider society
- apply mathematical skills relevant to computer science

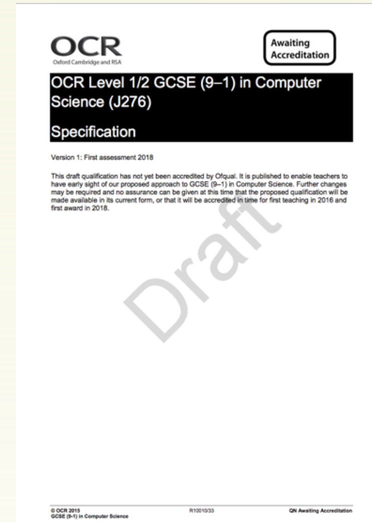
Subject content

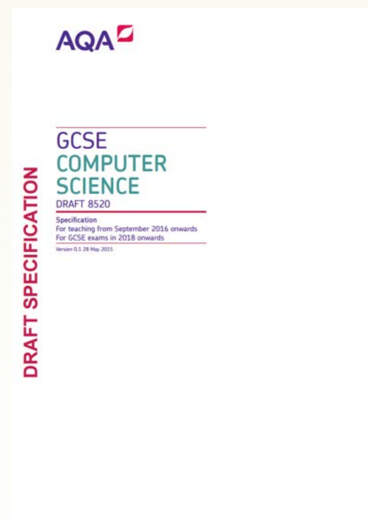
Knowledge and understanding

4. GCSE specifications must require students to develop a knowledge and understanding of the fundamentals of computer science and programming including:

3







DfE, 2015

Component 1: Practical programming

Scenario 2 – Computer Gaming Application

A local primary school has noticed that many of its pupils are playing computer games in their spare time. The school thinks that this may be a way that they can help pupils to learn.

You have been asked to develop a computer game that could be used to help teach 7 – 11 year old pupils in the classroom.

The school is looking at building a range of games in different subjects.

You must pick ONE subject area from the list below that your game will help to teach:

- Mathematics
- English
- Science
- ICT

Your game must teach one area of the subject:

- Mathematics – e.g. a game that will teach children to add numbers together
- English – e.g. a game that will teach children how to build a sentence
- Science – e.g. a game to teach children health and safety in the laboratory
- ICT – e.g. a game that will teach children to identify the different parts of a computer

Key Stage 5

16-18 years old

Department for Education

GCE AS and A level subject content for computer science

Introduction

1. AS and A level subject content sets out the knowledge, understanding and skills common to all AS and A level specifications in computer science.

Aims and objectives

2. All specifications in computer science must build on the knowledge, understanding and skills established at key stage 4 and encourage students to develop a broad range of the knowledge, understanding and skills of computing, as a basis for progression into further learning and/or employment.

3. AS and A level specifications in computer science must encourage students to develop:

- an understanding of, and the ability to apply, the fundamental principles and concepts of computer science, including abstraction, decomposition, logic, algorithms and data representation
- the ability to analyse problems in computational terms through practical experience of solving such problems, including writing programs to do so
- the capacity for thinking creatively, innovatively, analytically, logically and critically
- the capacity to see relationships between different aspects of computer science
- mathematical skills (as set out in the attached annex)
- the ability to articulate the individual (moral), social (ethical), legal and cultural opportunities and risks of digital technology

Subject content

Knowledge and understanding

4. AS and A level specifications must require students to develop a knowledge and understanding of the fundamentals of computer science and programming including:

- fundamentals of programming
- the concept of data type, including primitive data types and complex data structures
- data representation

Published: April 2014

1 2 In a functional programming language, a recursively defined function named `map` and a function named `double` are defined as follows:

```
map f []     = []
map f (x:xs) = f x : map f xs

double x     = 2 * x
```

The function `map` has two parameters, a function `f`, and a list that is either empty (indicated as `[]`), or non-empty, in which case it is expressed as `(x:xs)` in which `x` is the head and `xs` is the tail, which is itself a list.

1 2 . 1 In **Table 6**, write the value(s) that are the head and tail of the list `[ 1, 2, 3, 4 ]`.

[1 mark]

Table 6

Head	
Tail	

The result of making the function call `double 3` is `6`.

1 2 . 2 Calculate the result of making the function call listed in **Table 7**.

[1 mark]

Table 7

Function Call	Result
`map double [ 1, 2, 3, 4 ]`	

1 2 . 3 Explain how you arrived at your answer to question **1 2 . 2** and the recursive steps that you followed.

[3 marks]

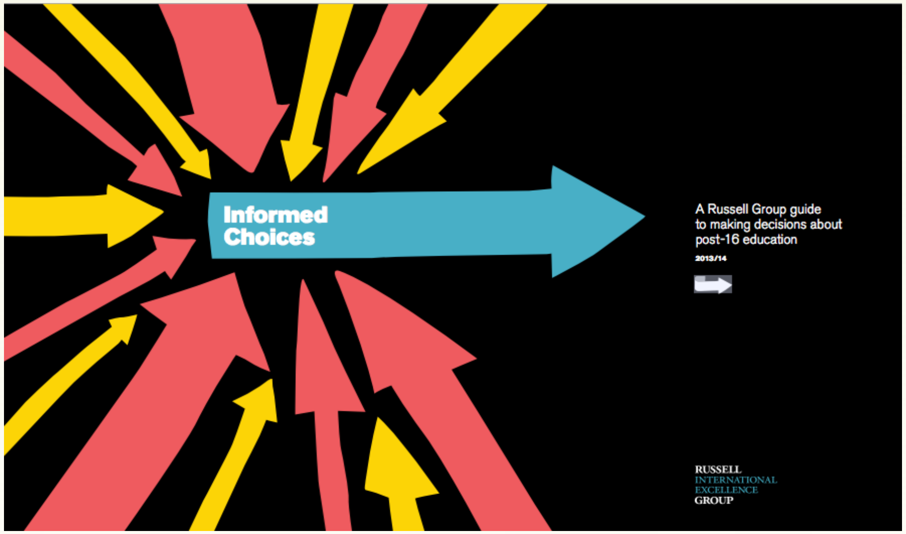
AQA, 2015,
qv AQA, 2015

2.1 A Complex problem

A Complex problem is one that has the potential to involve one or more of the following to the depth indicated in columns 2 and 3 of Table 1, below, when *automated*.

- Non-trivial algorithms, standard or user-defined, e.g. a graph traversal algorithm, recursive algorithms
- Use of sophisticated features of programming language / complexity of programming language, e.g. sophisticated data structures, runtime created objects, user-defined OOP classes
- Time-based simulation
- Development of program solutions for portable devices / games consoles
- Complexity of non-computing field of the problem, e.g. 3-D vector manipulation
- Communication Protocols, e.g. TCP connections
- Image Processing / pattern recognition, e.g. steganography, use of regular expressions

University entrance



Computer Science

ESSENTIAL ADVANCED LEVEL QUALIFICATIONS

For some courses, Mathematics. For some courses Computing/ Computer Science.

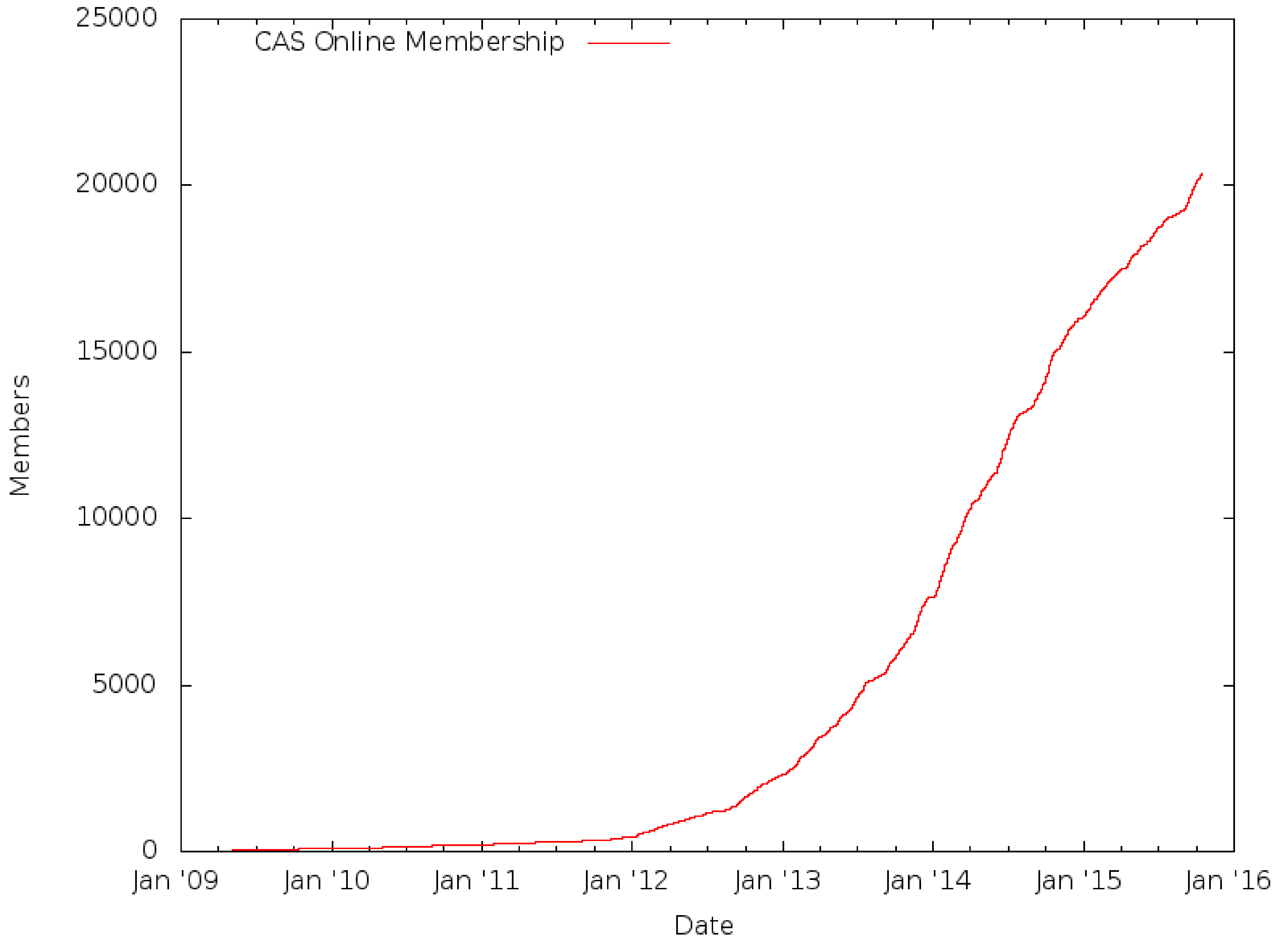
USEFUL ADVANCED LEVEL QUALIFICATIONS

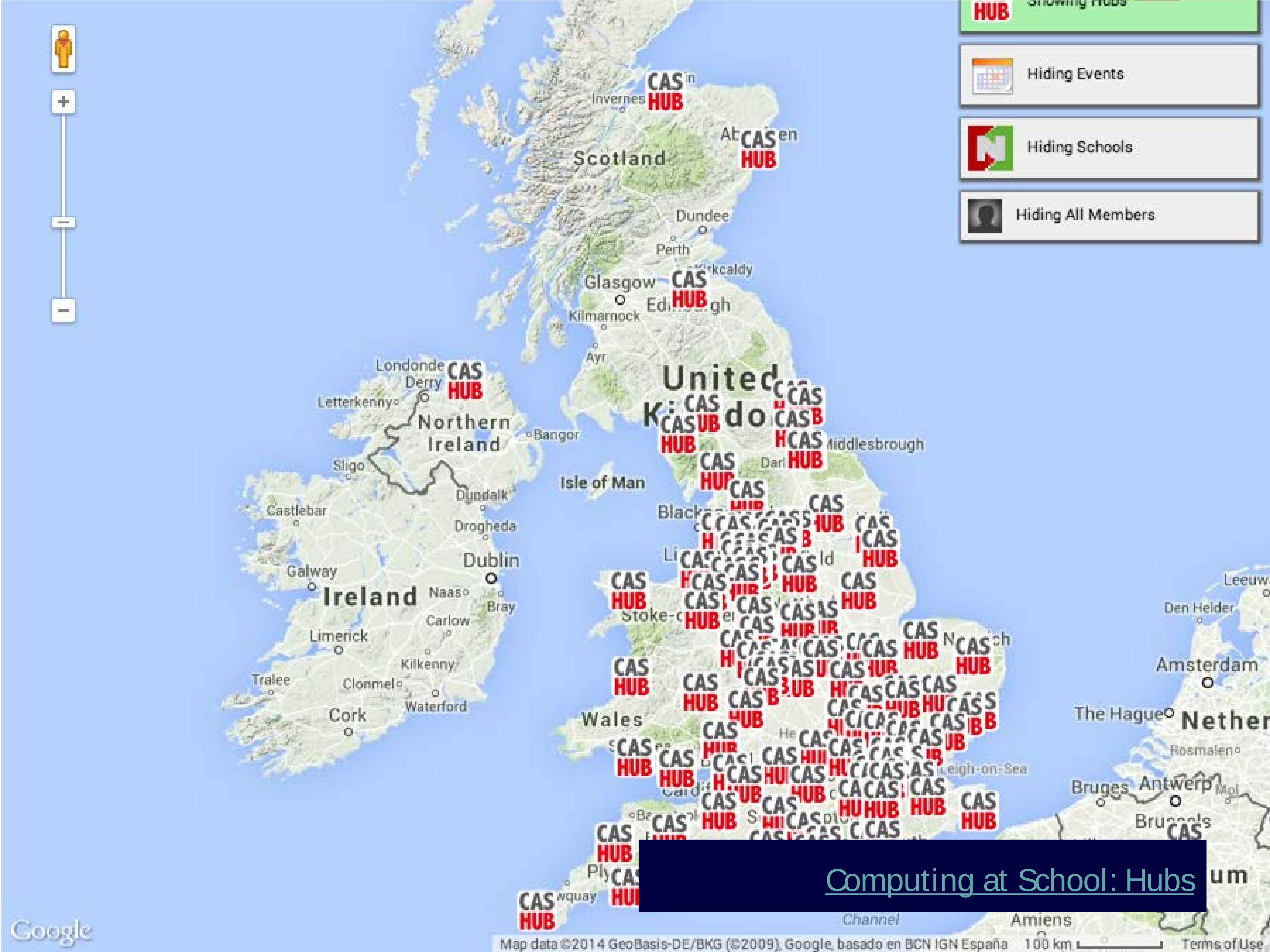
Mathematics, Further Mathematics, Computing/Computer Science, Physics, Philosophy, ICT.

Useful for:

Aeronautical engineering
Biochemistry
Biology
Chemical engineering
Economics
Chemistry
Civil engineering
Geology / Earth sciences
Electrical / Electronic engineering
Engineering
Mathematics
Mechanical engineering
Medicine
Materials science
Optometry
Orthoptics
Pharmacy
Physics
Psychology
Sociology
Teacher training (!)

With a little help from our friends

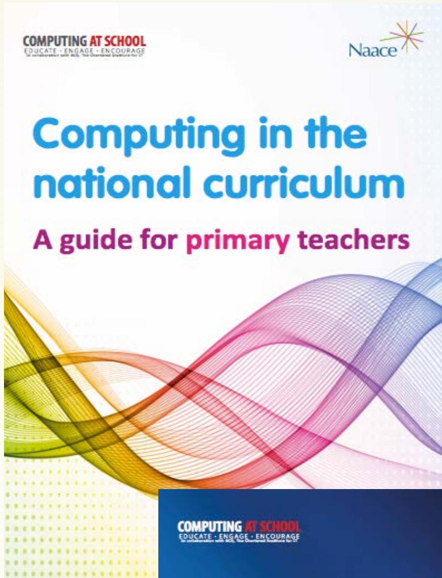

CAS Online Membership
Members
Date
25000
20000
15000
10000
5000
0
Jan '09
Jan '10
Jan '11
Jan '12
Jan '13
Jan '14
Jan '15
Jan '16

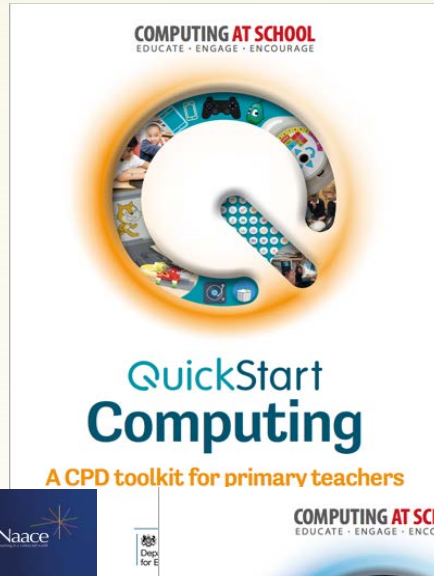


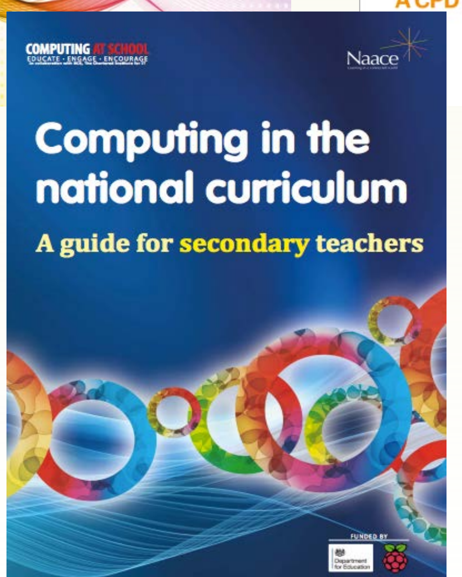
Computing at School: Hubs

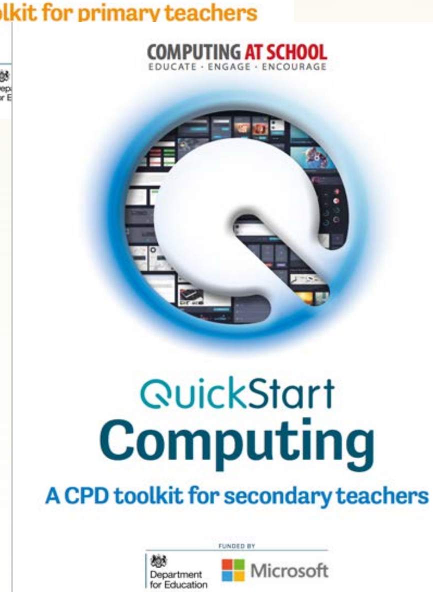
[Computing at School: Master Teachers]

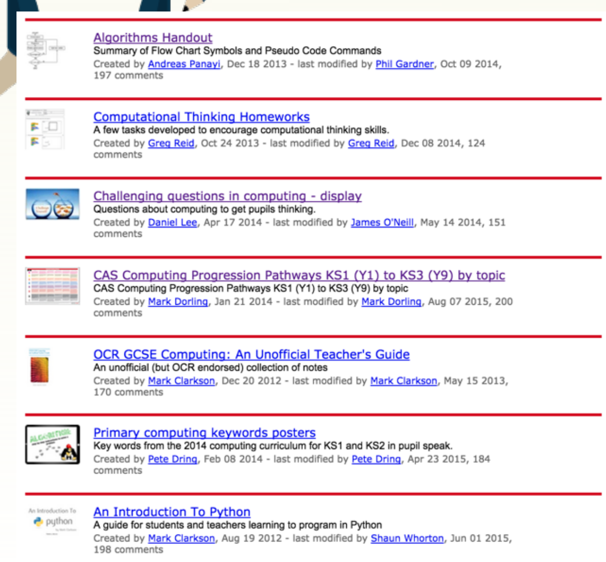
CAS resources











Algorithms Handout
Summary of Flow Chart Symbols and Pseudo Code Commands
Created by Andreas Panayi, Dec 18 2013 - last modified by Phil Gardner, Oct 09 2014, 197 comments

Computational Thinking Homeworks
A few tasks developed to encourage computational thinking skills.
Created by Greg Reid, Oct 24 2013 - last modified by Greg Reid, Dec 08 2014, 124 comments

Challenging questions in computing - display
Questions about computing to get pupils thinking.
Created by Daniel Lee, Apr 17 2014 - last modified by James O'Neill, May 14 2014, 151 comments

CAS Computing Progression Pathways KS1 (Y1) to KS3 (Y9) by topic
CAS Computing Progression Pathways KS1 (Y1) to KS3 (Y9) by topic
Created by Mark Dorling, Jan 21 2014 - last modified by Mark Dorling, Aug 07 2015, 200 comments

OCR GCSE Computing: An Unofficial Teacher's Guide
An unofficial (but OCR endorsed) collection of notes
Created by Mark Clarkson, Dec 20 2012 - last modified by Mark Clarkson, May 15 2013, 170 comments

Primary computing keywords posters
Key words from the 2014 computing curriculum for KS1 and KS2 in pupil speak.
Created by Pete Dring, Feb 08 2014 - last modified by Pete Dring, Apr 23 2015, 184 comments

An Introduction To Python
A guide for students and teachers learning to program in Python
Created by Mark Clarkson, Aug 19 2012 - last modified by Shaun Whorton, Jun 01 2015, 198 comments

Apps for Good

YOUNG REWIRED STATE

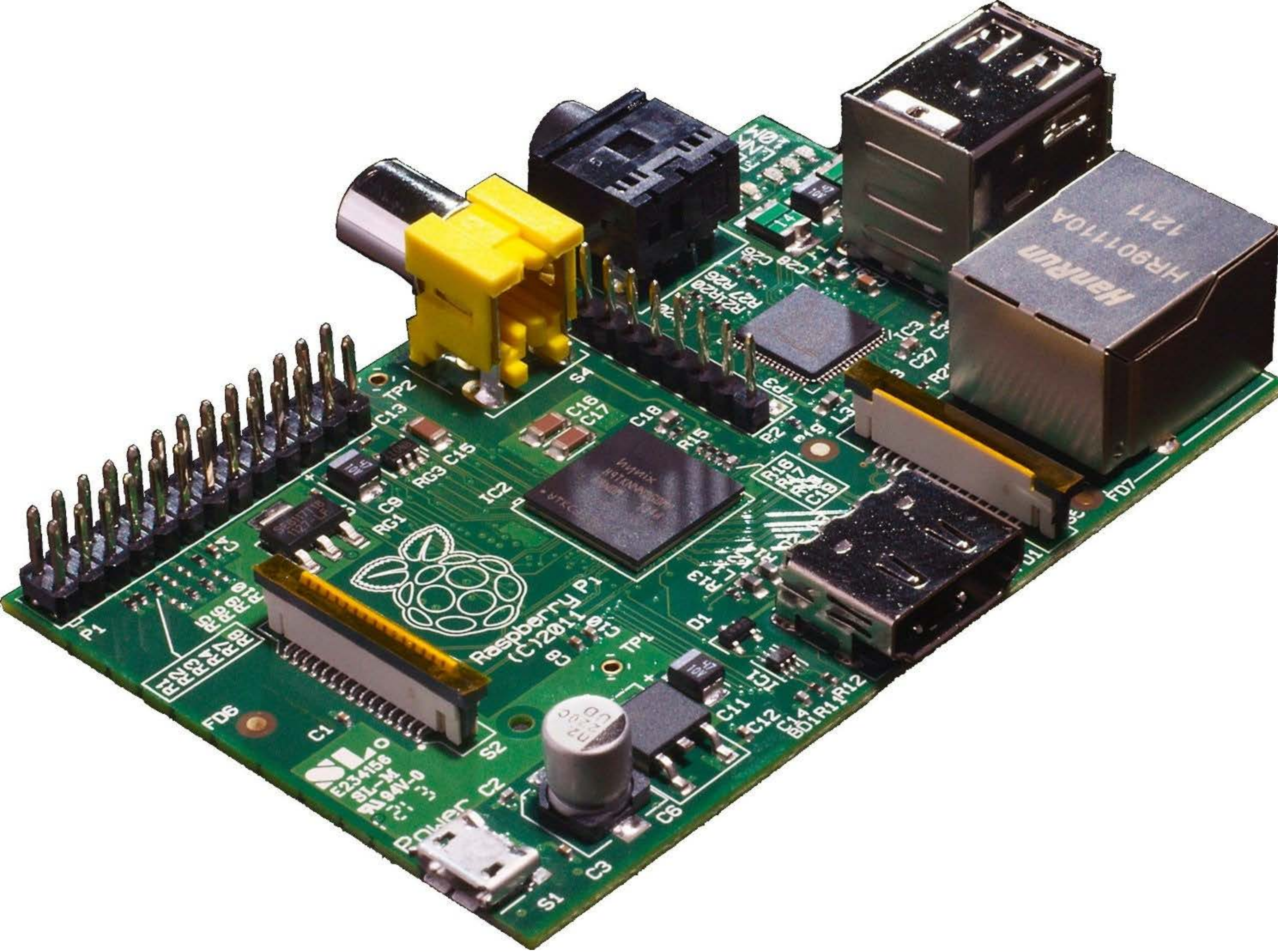
3D ami

digital schoolhouse

Excellence in ICT & Computer Science

{CODE CLUB}

ocado


Raspberry Pi
(C)2011
HanRun
HR901110A
1211
Power
P1
P2
P3
FD6
FD7
S1
S2
S4
IC2
TP1
TP2
RG1
RG3
C1
C2
C3
C6
C9
C11
C12
C13
C15
C16
C17
C18
D1
E234156
94V-0

MAKE IT DIGITAL

NEW

BBC Make It Digital: Get creative with coding and digital technology

What it's all about
Will.i.am on coding

Be careful what you wish for...

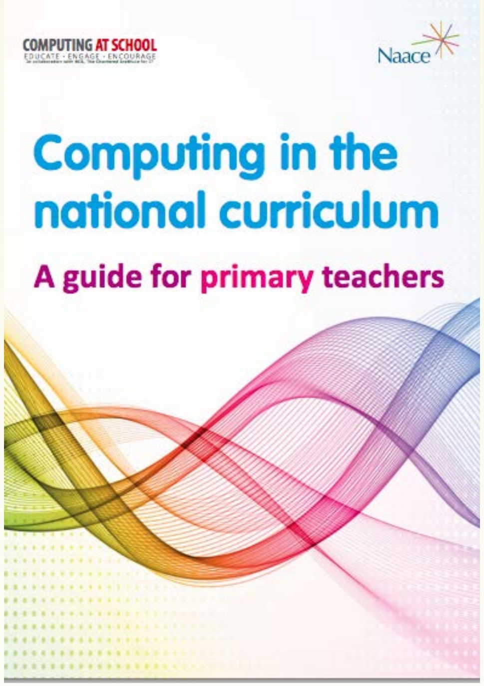
Transition

DfE:
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.

Ofsted:
The statutory requirement from 1 September 2014 is for maintained schools to teach the relevant national curriculum programmes of study by the end of the key stage.

Why programming?

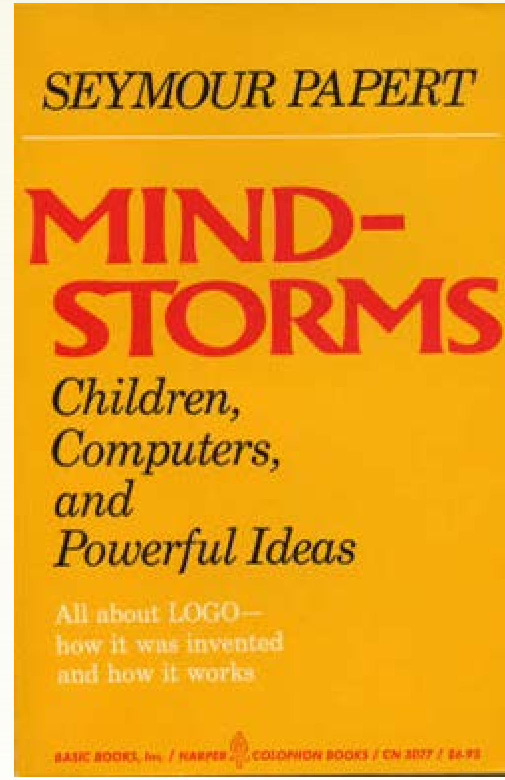
the practical experience of programming, [is] almost certainly the best way for primary pupils to learn about computer science



Berry, 2013

Programming develops thinking

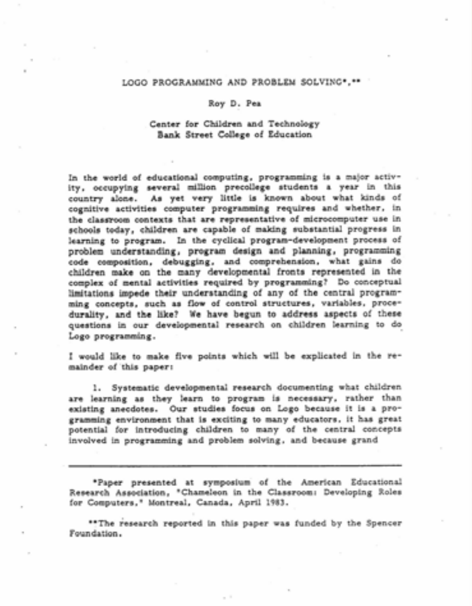
I began to see how children who had learned to program computers could use very concrete computer models to think about thinking and to learn about learning and in doing so, enhance their powers as psychologists and as epistemologists.



Papert, 1980

But teach problem solving too

We find that the entry level of Logo does not present conceptual problems for the school-aged child... With accompanying instruction in thinking skills, developments in planning skill may in fact be achieved.



LOGO PROGRAMMING AND PROBLEM SOLVING*,**

Roy D. Pea

Center for Children and Technology
Bank Street College of Education

In the world of educational computing, programming is a major activity, occupying several million precollege students a year in this country alone. As yet very little is known about what kinds of cognitive activities computer programming requires and whether, in the classroom contexts that are representative of microcomputer use in schools today, children are capable of making substantial progress in learning to program. In the cyclical program-development process of problem understanding, program design and planning, programming code composition, debugging, and comprehension, what gains do children make on the many developmental fronts represented in the complex of mental activities required by programming? Do conceptual limitations impede their understanding of any of the central programming concepts, such as flow of control structures, variables, procedurality, and the like? We have begun to address aspects of these questions in our developmental research on children learning to do Logo programming.

I would like to make five points which will be explicated in the remainder of this paper:

1. Systematic developmental research documenting what children are learning as they learn to program is necessary, rather than existing anecdotes. Our studies focus on Logo because it is a programming environment that is exciting to many educators, it has great potential for introducing children to many of the central concepts involved in programming and problem solving, and because grand

*Paper presented at symposium of the American Educational Research Association, "Chameleon in the Classroom: Developing Roles for Computers," Montreal, Canada, April 1983.

**The research reported in this paper was funded by the Spencer Foundation.

Pea, 1983

In summary

It’s not about the coding

It’s easier to read code than to write code

It’s easier to edit code that to start from a blank screen

Look for interesting contexts

Making things matters

Pair programming is powerful

Debugging helps grow mindsets

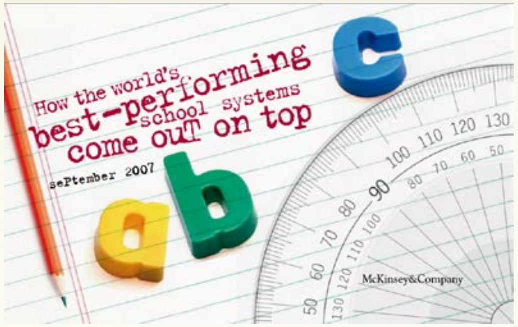
Go for depth not breadth

Nurture curiosity

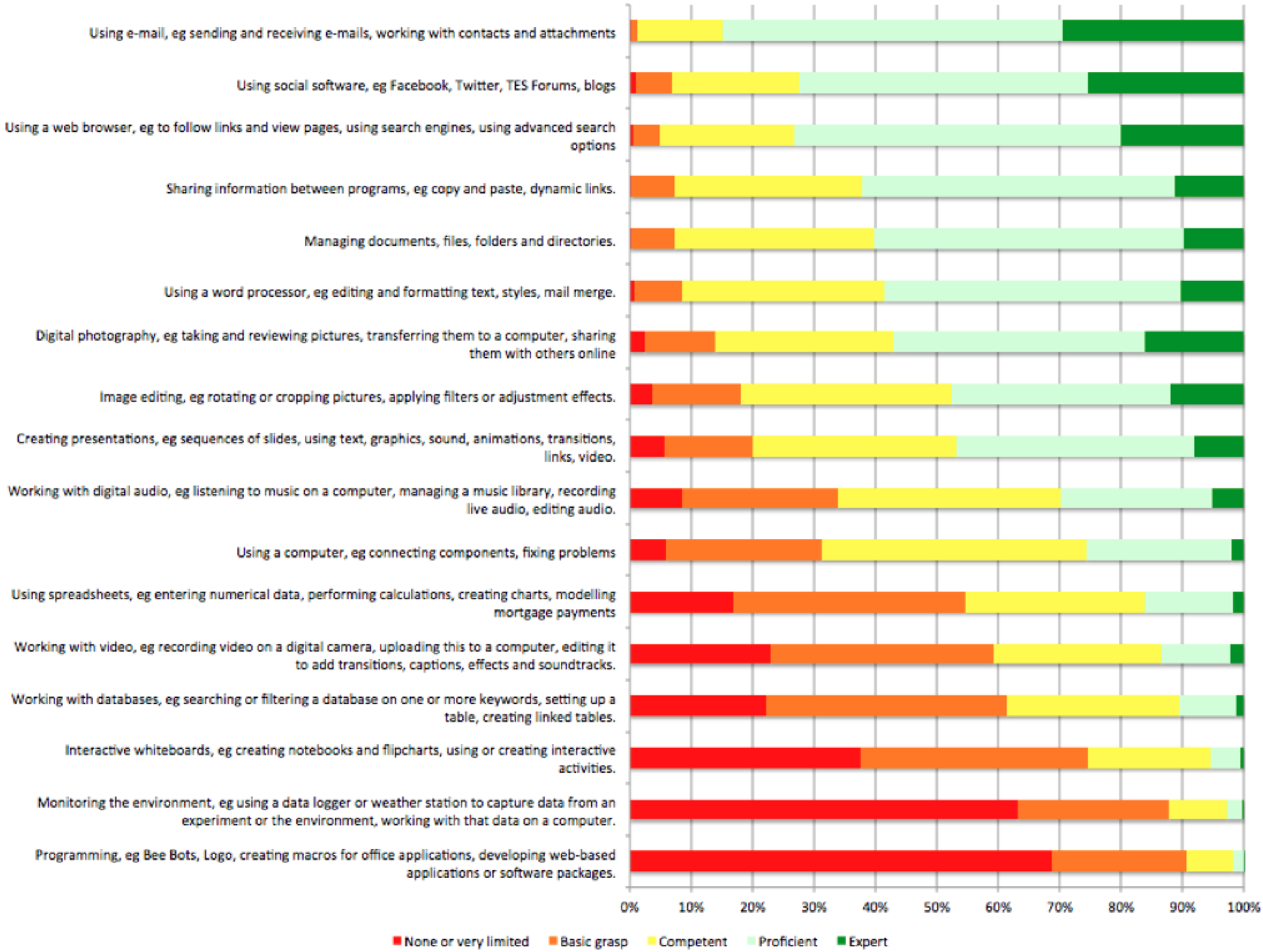
This is for everyone

Teaching matters

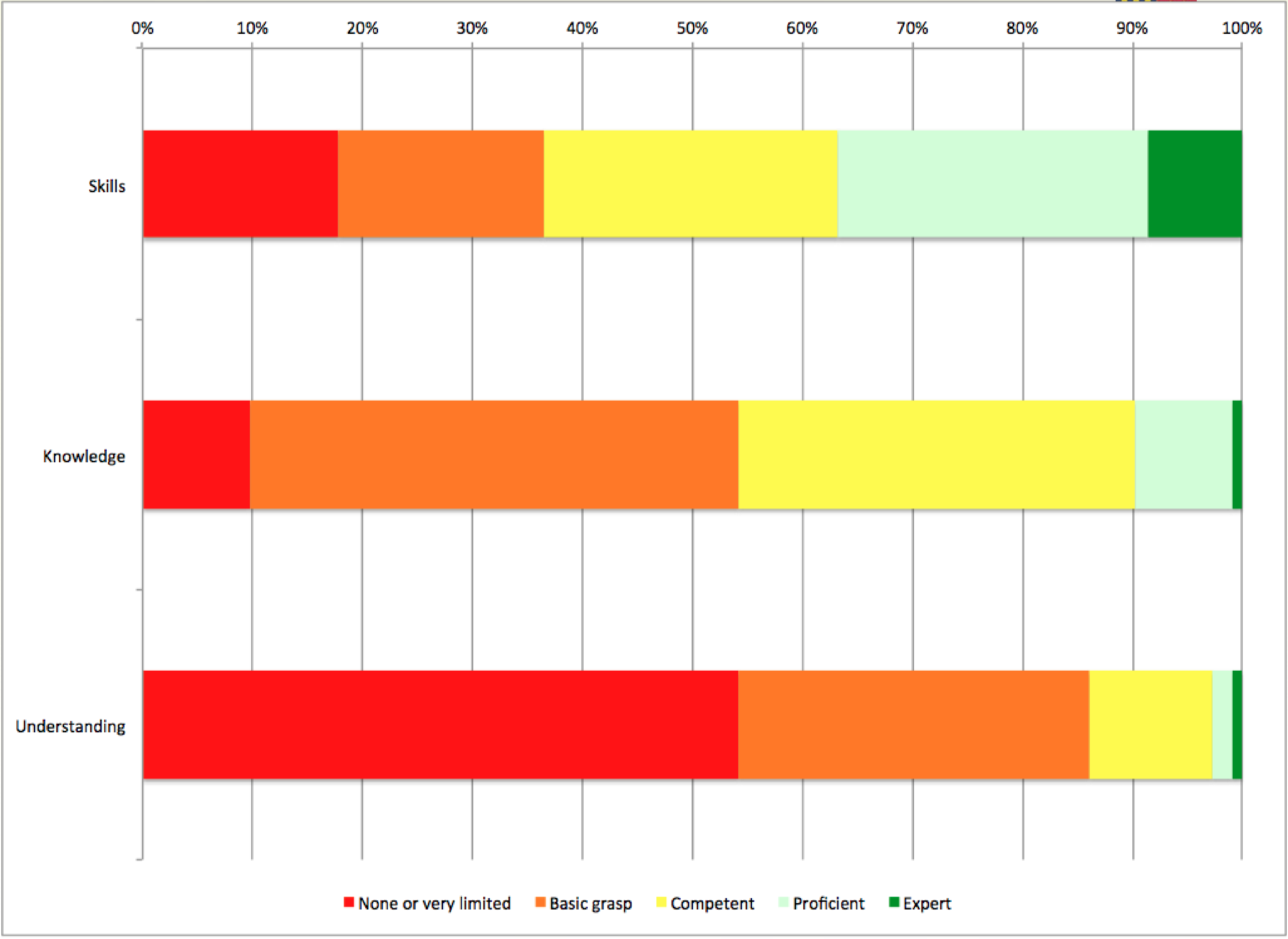
The quality of an education system cannot exceed the quality of its teachers

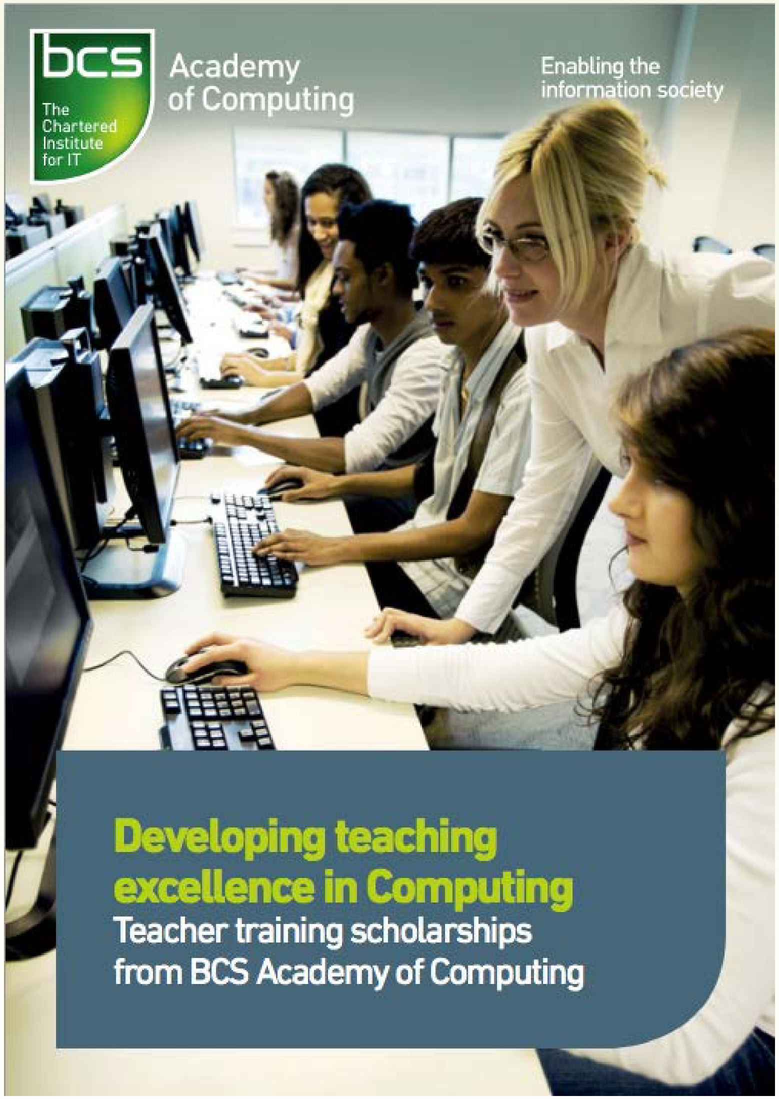


Interview South Korea 2007, cited in [Barber and Mourshed 2007]



Using e-mail, eg sending and receiving e-mails, working with contacts and attachments
Using social software, eg Facebook, Twitter, TES Forums, blogs
Using a web browser, eg to follow links and view pages, using search engines, using advanced search options
Sharing information between programs, eg copy and paste, dynamic links.
Managing documents, files, folders and directories.
Using a word processor, eg editing and formatting text, styles, mail merge.
Digital photography, eg taking and reviewing pictures, transferring them to a computer, sharing them with others online
Image editing, eg rotating or cropping pictures, applying filters or adjustment effects.
Creating presentations, eg sequences of slides, using text, graphics, sound, animations, transitions, links, video.
Working with digital audio, eg listening to music on a computer, managing a music library, recording live audio, editing audio.
Using a computer, eg connecting components, fixing problems
Using spreadsheets, eg entering numerical data, performing calculations, creating charts, modelling mortgage payments
Working with video, eg recording video on a digital camera, uploading this to a computer, editing it to add transitions, captions, effects and soundtracks.
Working with databases, eg searching or filtering a database on one or more keywords, setting up a table, creating linked tables.
Interactive whiteboards, eg creating notebooks and flipcharts, using or creating interactive activities.
Monitoring the environment, eg using a data logger or weather station to capture data from an experiment or the environment, working with that data on a computer.
Programming, eg Bee Bots, Logo, creating macros for office applications, developing web-based applications or software packages.
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
None or very limited
Basic grasp
Competent
Proficient
Expert


0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Skills
Knowledge
Understanding
None or very limited
Basic grasp
Competent
Proficient
Expert



National College for Teaching & Leadership

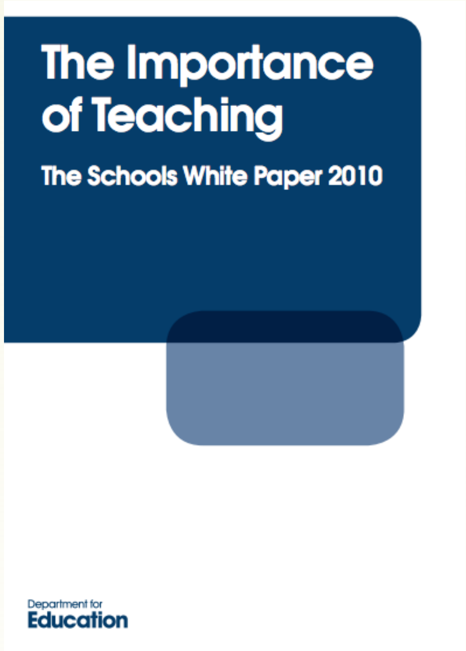
Initial teacher training (ITT): training bursary guide

Academic year 2014 to 2015 (Version 1.3)

September 2014

CPD matters

We know that teachers learn best from other professionals and that an ‘open classroom’ culture is vital: **having the opportunity to plan, prepare, reflect and teach with other teachers**... [and yet] two-thirds of all professional development is ‘passive learning’ – sitting and listening to a presentation.



DfE, 2010

Some CAS research questions

How should we teach programming to children?
How effective are "unplugged" approaches to learning computer science?
How should we assess computing?
Does an early education in computing improve outcomes in Maths or English?
How can computational thinking skills enrich learning in maths, science, and other subjects?
In what order are computational concepts best learned?
Women are massively under-represented in computing. What practical strategies help?


THIS IS FOR EVERYONE

Questions?

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

These slides: bit.ly/erte15