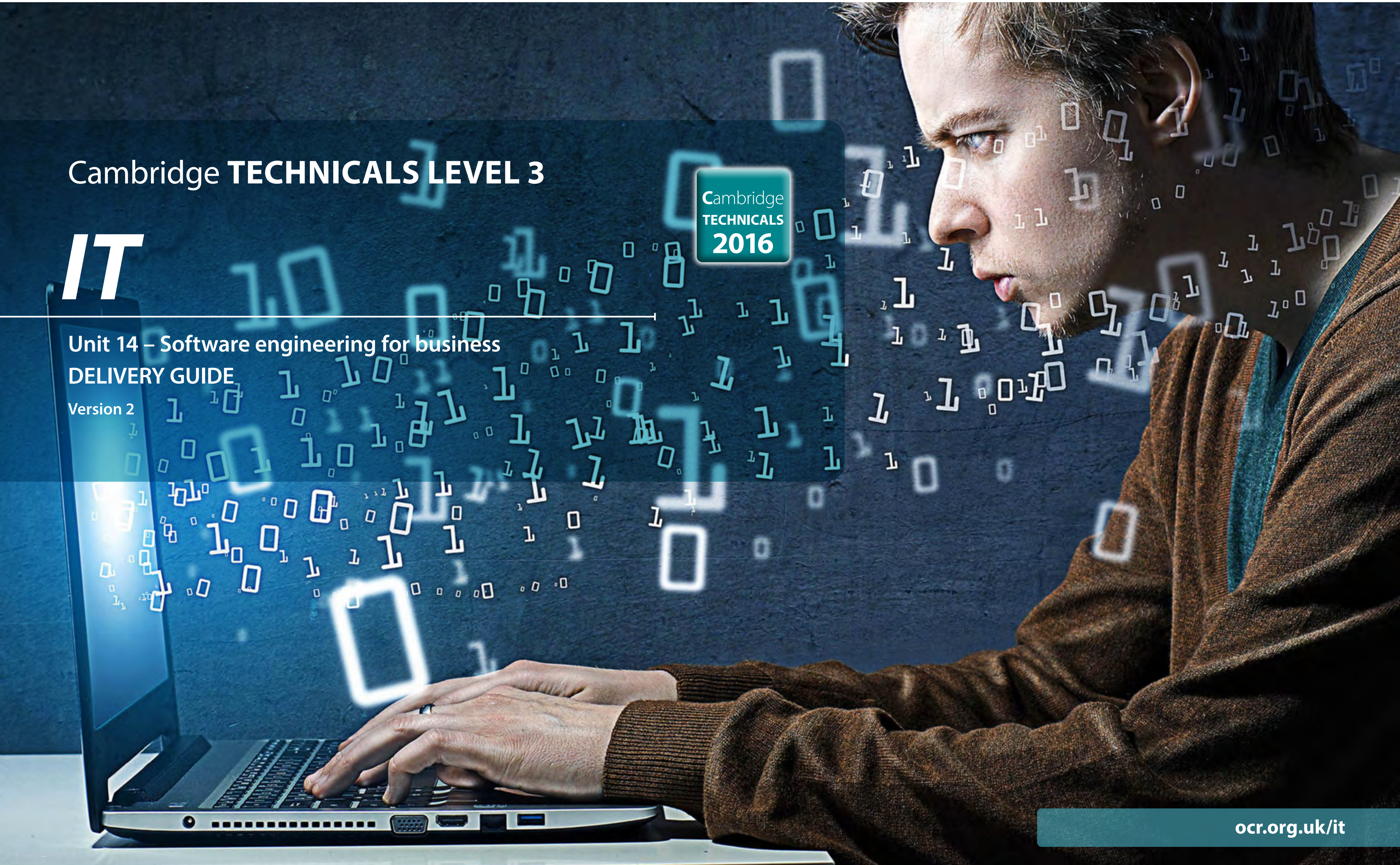
OCR
Oxford Cambridge and RSA



Cambridge **TECHNICALS LEVEL 3**

IT

Cambridge TECHNICALS 2016

Unit 14 – Software engineering for business

DELIVERY GUIDE

Version 2

ocr.org.uk/it

CONTENTS

Introduction 3

Related Activities 4

Key Terms 6

Misconceptions 7

Suggested Activities:

- Learning Outcome (LO1) 9
- Learning Outcome (LO2) 15
- Learning Outcome (LO3) 19
- Learning Outcome (LO4) 21

INTRODUCTION

This Delivery Guide has been developed to provide practitioners with a variety of creative and practical ideas to support the delivery of this qualification. The Guide is a collection of lesson ideas with associated activities, which you may find helpful as you plan your lessons.

OCR has collaborated with current practitioners to ensure that the ideas put forward in this Delivery Guide are practical, realistic and dynamic. The Guide is structured by learning outcome so you can see how each activity helps you cover the requirements of this unit.

We appreciate that practitioners are knowledgeable in relation to what works for them and their learners. Therefore, the resources we have produced should not restrict or impact on practitioners' creativity to deliver excellent learning opportunities.

Whether you are an experienced practitioner or new to the sector, we hope you find something in this guide which will help you to deliver excellent learning opportunities.

If you have any feedback on this Delivery Guide or suggestions for other resources you would like OCR to develop, please email resources.feedback@ocr.org.uk.

OPPORTUNITIES FOR ENGLISH AND MATHS SKILLS DEVELOPMENT AND WORK EXPERIENCE

We believe that being able to make good progress in English and maths is essential to learners in both of these contexts and on a range of learning programmes. To help you enable your learners to progress in these subjects, we have signposted opportunities for English and maths skills practice within this resource. We have also identified any potential work experience opportunities within the activities. These suggestions are for guidance only. They are not designed to replace your own subject knowledge and expertise in deciding what is most appropriate for your learners.

ABC English   123 Maths   Work

Please note

The timings for the suggested activities in this Delivery Guide **DO NOT** relate to the Guided Learning Hours (GLHs) for each unit.

Assessment guidance can be found within the Unit document available from www.ocr.org.uk.

The latest version of this Delivery Guide can be downloaded from the OCR website.

UNIT AIM

The aim of this unit is to give you practical experience of writing computer programs for specific requirements, such as those found in a business. Programmers' first jobs tend to be coding against specific requirements or maintaining an existing program. You need only a basic appreciation of the full system life cycle at this level, freeing up more time for practical programming experience.

This unit focuses on developing code for a single customer with specific requirements. If you are on the application developer pathway, you could follow this unit with the games design and prototyping unit, which focuses on developing for a mass market, allowing you to further develop your coding skills and experience.

This unit is optional within the Applications Developer and Data Analyst specialist pathways. Application developers create, test and program applications software and therefore use the knowledge skills and understanding associated with software engineering. Data analysts require a good understanding of computers and software and an insight into software engineering can assist them in their design and analysis of systems and data.

Unit 14 Software engineering for business	
LO1	Understand universal programming constructs
LO2	Be able to investigate business requirements for programming solutions
LO3	Be able to develop software solutions to meet business requirements
LO4	Be able to propose software solutions to meet business requirements

To find out more about this qualification please go to: http://www.ocr.org.uk/qualifications/cambridge-technicals-it-level-3-certificate-extended-certificate-introductory-diploma-foundation-diploma-diploma-05838-05842-2016-suite

Cambridge TECHNICALS 2016

2016 Suite

- **New suite for first teaching September 2016**
- **Externally assessed content**
- **Eligible for Key Stage 5 performance points from 2018**
- **Designed to meet the DfE technical guidance**

RELATED ACTIVITIES

The Suggested Activities in this Delivery Guide listed below have also been related to other Cambridge Technicals in IT units/Learning Outcomes (LOs). This could help with delivery planning and enable learners to cover multiple parts of units.

This unit (Unit 14)	Title of suggested activity	Other units/LOs	
LO1	Hello world Integers and floating point data types Booleans Strings and loops – Hangman part 1 Encapsulation – Hangman part 2 Maths game Calculator Times table production Input verifier	Unit 1 Fundamentals of IT	LO1 Understand computer hardware LO2 Understand computer software
		Unit 15 Games design and prototyping	LO3 Be able to develop game prototypes
LO2	Kate's Pizza	Unit 1 Fundamentals of IT	LO3 Understand business IT systems
		Unit 2 Global information	LO3 Understand the use of global information and the benefits to individuals and organisations LO4 Understand the legal and regulatory framework governing the storage and use of global information
		Unit 6 Application design	LO2 Be able to investigate potential solutions for application developments
		Unit 9 Product development	LO2 Be able to design products that meet identified client requirements
		Unit 11 Systems analysis and design	LO2 Be able to use investigative techniques to establish requirements for business systems
		Unit 12 Mobile Technology	LO3 Be able to determine solutions for the use of mobile technologies
		Unit 21 Web design prototyping	LO2 Be able to plan the development of an interactive website for an identified client
	Petrol pumps Stock control Prioritise requirements Requirements capture Finished document	Unit 1 Fundamentals of IT	LO3 Understand business IT systems
		Unit 2 Global information	LO3 Understand the use of global information and the benefits to individuals and organisations
		Unit 6 Application design	LO2 Be able to investigate potential solutions for application developments
		Unit 9 Product development	LO2 Be able to design products that meet identified client requirements
		Unit 11 Systems analysis and design	LO2 Be able to use investigative techniques to establish requirements for business systems
		Unit 12 Mobile Technology	LO3 Be able to determine solutions for the use of mobile technologies
		Unit 21 Web design prototyping	LO2 Be able to plan the development of an interactive website for an identified client

This unit (Unit 14)	Title of suggested activity	Other units/LOs	
LO3	Designing for a software engineer that isn't you	Unit 6 Application design	LO1 Be able to generate designs for application solutions
		Unit 11 Systems analysis and design	LO3 Be able to develop and document models for business systems
		Unit 15 Games design and prototyping	LO2 Be able to develop game concepts
	Adapt to enhance	Unit 1 Fundamentals of IT	LO4 Understand employability and communication skills used in an IT environment
		Unit 15 Games design and prototyping	LO2 Be able to develop game concepts LO3 Be able to develop game prototypes
	Third party code	Unit 15 Games design and prototyping	LO3 Be able to develop game prototypes
	Coding	Unit 15 Games design and prototyping	LO3 Be able to develop game prototypes
	Would you do it differently if you did it again?	Unit 8 Project management	LO4 Be able to carry out project evaluations
	Testing, testing	Unit 9 Product development	LO3 Be able to implement and test products
		Unit 15 Games design and prototyping	LO3 Be able to develop game prototypes
		Unit 21 Web design and prototyping	LO3 Be able to create prototype websites for an identified client
LO4	I have to do WHAT? How bad can it be? Present to different audiences Practise, practise, practise Smile, you're on camera Understand then analyse Maintenance – creating version 2	Unit 1 Fundamentals of IT	LO4 Understand employability and communication skills used in an IT environment
		Unit 6 Application design	LO4 Be able to present application solutions to meet client and user requirements
		Unit 7 Data analysis and design	LO4 Be able to present data analysis and design solutions to stakeholders
		Unit 10 Business computing	LO4 Be able to present data analysis outcomes
		Unit 15 Games design and prototyping	LO4 Be able to present and evaluate game concepts
		Unit 22 Big Data analytics	LO3 Be able to provide information resulting from processing Big Data

KEY TERMS

Explanations of the key terms used within this unit, in the context of this unit	
Key term	**Explanation**
Compiler	The tool used to convert an entire high-level code listing into an executable program before it is run. See Translator and Interpreter.
Constructive feedback	Feedback given that describes ways in which the work could be improved. See Positive feedback.
Executable code	The sequence of instructions that a processor can run. This is usually non-readable by humans and is the result of parsing source code through a translator.
Graphical User Interface (GUI)	Graphical User Interface (GUI) denotes a pointer-driven interface that a user operates in order to input data and commands into a computer program. The usual alternative is a text-based command line user interface where the user must type in commands on a keyboard.
High-level code	Series of instructions written in a language that is closer to humanly-understandable language than a low-level language, but a processor will not be able to run the commands without first using a translator.
Integrated Development Environment (IDE)	A collection of tools to assist in the creation of a software product. Tools such as a GUI designer, code editor and debugger.
Interpreter	The tool used to convert lines of a high-level code listing into an executable program, one line at a time, and execute them before moving onto the next line. See Compiler and Translator.
Low-level code	Series of instructions written in a language that is closer to processor-specific instructions than a humanly-understandable language.
Positive feedback	Feedback given that encourages further work in the same direction. Negative feedback would be discouraging further similar work. See Constructive feedback.
Programmer	A person focussing specifically on writing lines of code in the development phase of the project life cycle. See Software engineer.
Project life cycle	Refers to the lifetime of a software product from conception to maintenance. There are different ways of approaching the project life cycle used within the software industry.
Software engineer	A person able to carry out all phases of the project life cycle.
Source code	The program listing that is written by humans but cannot be directly run by a processor. A Translator will convert the source code into executable code.
Syntax	The strict rules governing the composition of lines of code.
Translator	The tool that compilers and interpreters use to convert lines of high-level code listing into executable instructions for a processor. See Compiler and Interpreter.

MISCONCEPTIONS

Some common misconceptions and guidance on how they could be overcome

What is the misconception?	How can this be overcome?	Resources which could help
You have to be good at maths in order to be a programmer	You have to be able to visualise a problem in order to solve it, but the actual mathematical solving of a problem is left to the computer following the subsequent program. A fear of maths can be overcome with any simple example, such as programming Pythagoras where students must understand the concept but not have to perform complex square roots in their heads.	www.mathsisfun.com/pythagoras.html
I didn't do computing at school so I won't be able to do this unit	Reassurance that this is completely wrong coupled with a simple start in programming that contains early successes for the learners to demonstrate to themselves they can do this. For example, write a program to show 'Hello World' on the screen, changing the code to show 'Hello [their name]' to prove they can change a program to make different outcomes happen.	http://blogs.msdn.com/b/alfredth/archive/2006/11/02/what-is-so-scary-about-programming.aspx Computer Coding – What most schools don't teach www.youtube.com/watch?v=nPblG6ceqOs Inspiring video showing how successful people like Bill Gates, Mark Zuckerberg and Will.I.Am started out when they were learning programming. You Should Learn to Program: Christian Genco at TEDxSMU www.youtube.com/watch?v=xfBWk4nw440 Funny and inspiring video showing how the world is changing faster than people are. Links to further TED videos.
We should be learning to program in the trendy language	It is unrealistic to believe you will use only one language for an entire career. Far better to learn transferable programming constructs in a language carefully selected by your instructors for any number of reasons, with familiarity being a very strong reason. Instructors must be able to confidently guide and spot problems in order to properly assist learners. Large banks of resources take time to convert to new languages. A visual and basic language is more appropriate for learners of all abilities than a complex language that only the most able will grasp with ease.	www.tiobe.com/index.php/content/paperinfo/tpci/index.html TIOBE index of the world's most popular programming languages.

Some common misconceptions and guidance on how they could be overcome		
What is the misconception?	**How can this be overcome?**	**Resources which could help**
I will never be able to stand in front of an audience and present	This is a complex issue, especially with certain learning related to medical conditions which are in conflict with presentation skills such as eye contact and conversing with strangers. This can be tackled with confidence boosting and demonstrating that since others with the condition have done it, they too can do it (see resources links). Fear is common in everybody, including industry leaders, so it is not surprising that learners unaccustomed to public speaking will be reluctant. Fear can be overcome with frequent and simple presentation opportunities, such as in lesson 1 each learner introduces the learner sat next to them to the group after briefly conversing with them. Practising trivial presentations with cue-cards and positive feedback should build confidence. Ultimately, it is the act of doing the presentation rather than the quality of presentation that is being assessed.	www.businessinsider.com/10-biggest-public-speaking-phobias-2011-2 This link contains further links to help with specific related phobias. www.anxietycoach.com/fear-of-public-speaking.html Hints and tips. http://autismmythbusters.com/general-public/famous-autistic-people/ Confidence boosting. https://whyiwonttalk.wordpress.com/tag/public-speaking/ Relate to others in the same position. www.jkp.com/uk/an-asperger-s-guide-to-public-speaking.html

SUGGESTED ACTIVITIES

LO No:	**1**		
LO Title:	**Understand universal programming constructs**		
Title of suggested activity	**Suggested activities**	**Suggested timings**	**Also related to**

Hello world	Learners must demonstrate using the programming constructs listed in the teaching content. This can be done as one major project; however it is recommended to start small and build on capability with multiple small example programs designed to demonstrate one concept each and to give the learners early successes with programming. Each example program builds from a single line of code to the finished program within one lesson, taking the learners through building a simple program rather than simply typing it in from a listing. Lesson-sized programs also make absentees easier to deal with upon their return.
	
	Teaching programming via a series of simple games is encouraged as it will capture the imagination of learners in the same way that the first home computers captured the imagination of schoolchildren in the '80s. It is important that learners are not just given a listing to type in. However, these examples must be built up line by line with challenges along the way.
	
	These introductory programs do not have to be games or visual, although such an approach would capture the imagination of the younger learners. The production of times tables could be used to teach fixed loops. Asking for a whole number of cup-cakes to be input can be used to teach data-type checking and conditional loops which repeat until the input is within allowable constraints. During these lessons the learners will have built up a bank of example programs which they can draw from when giving examples of the programming constructs listed in the Teaching Content.
	
	Start with the initial 'Hello world' style program which is the customary first program any programmer writes when working with a new environment. This program simply places the words 'Hello world!' on the screen. Learners could then change the message to show the connection between source code and running code. A simple 'if' statement could be built onto this program asking for age in years to be inputted and the program to state either 'Adult' or 'Child' in response. Further age ranges could be added to demonstrate a complex 'if' statement. This would cover the selection programming construct. In Microsoft Visual Basic.Net, the code could be:
	`Dim intAge As Integer = InputBox("Please enter age in years.")`
	`If intAge < 18 Then`
	`   MsgBox("You are a child.")`
	`Else`
	`   MsgBox("You are an adult.")`
	`End If`
	
	Learners could expand the 'if' statement to detect and show messages for likely errors such as negative ages or ages greater than 150.
		1 hour	Unit 1 LO1, LO2
			Unit 15 LO3

Title of suggested activity	Suggested activities	Suggested timings	Also related to

Integers and floating point data types	The above program most likely required an integer variable to be declared to allow for the age to be input. This can lead nicely into explaining the use of variables using the fundamental data types (introduction to fundamental data types and their use in example simple programs: https://en.wikibooks.org/wiki/A-level_Computing_2009/AQA/Problem_Solving,_Programming,_Data_Representation_and_Practical_Exercise/Fundamentals_of_Programming/Built-in_data_types).
	
	Investigating what happens if you try to input an age with a decimal place into an integer variable will lead into floating point data types. Remember the learners do not have to explain how the computer stores such data types; rather they must demonstrate competence with using them in a program.
	
	Activities might be discussions centred on simple programs to demonstrate basic programming constructs. For example, to demonstrate data types, learners could be given a program to type in which inputs two integer values and displays the total of the two numbers. In Microsoft Visual Basic.Net, the code could be:
	`Dim intNumber1 As Integer = InputBox("Please enter the first number.")`
	`Dim intNumber2 As Integer = InputBox("Please enter the second number.")`
	`Dim intResult As Integer = intNumber1 + intNumber2`
	`MsgBox(intResult)`
	
	Integer inputs will yield integer outputs, such as the inputs 5 and 7 will give the output 12. Learners can be encouraged to try real numbers as inputs to see what happens, such as 5.1 + 7.1. Learners can then modify the program to accept floating point numbers to allow decimal addition, such as the following example of Microsoft Visual Basic.Net code:
	`Dim sngNumber1 As Single = InputBox("Please enter the first number.")`
	`Dim sngNumber2 As Single = InputBox("Please enter the second number.")`
	`Dim sngResult As Single = sngNumber1 + sngNumber2`
	`MsgBox(sngResult)`
	
	Learners could try adding strings together which is most likely to be invalid input, depending on the language in use. For example; enter the word "hello" into the program as the first number. To allow string input, in Microsoft Visual Basic.Net, the code could be:
		30 minutes	Unit 1 LO1, LO2
			Unit 15 LO3

Title of suggested activity	Suggested activities	Suggested timings	Also related to

Integers and floating point data types (continued)	`Dim strNumber1 As String = InputBox(“Please enter the first number.”)`
	`Dim strNumber2 As String = InputBox(“Please enter the second number.”)`
	`Dim strResult As String = strNumber1 + strNumber2`
	`MsgBox(strResult)`
	
	Adding two strings together will show them concatenated, such as the inputs "Hello" and "World" would yield the output "HelloWorld". Learners can then see that the inputs 5 and 7 now yield "57" and not 12.
	
	As a task, learners could try to write a program which divides rather than adds the two input numbers. They can then see what happens when divide by zero is attempted. Code such as an 'if' statement could be added to prevent the divide by zero error.		

Booleans	Booleans can be explained with GUI objects such as check boxes and radio buttons. The whole class can be introduced to logic functions with kinaesthetic questioning such as: • 'Stand up if you are male OR female.' • 'Stand up if you are male AND female.' • 'Stand up if you are NOT female.' • 'Stand up if you are male and have hair below your shoulders.' • 'Stand up if you are NOT blonde.' • 'Stand up if you are blonde OR have blue eyes.' • 'Stand up if you are blonde AND have blue eyes.' • 'Stand up if you are NOT blonde AND do NOT have blue eyes.' A simple program can be created to demonstrate the use of Boolean logic to enforce rules for a bar: • 'You can only drink alcohol if you are NOT under 18 AND NOT driving.' • 'You are NOT allowed into the night club wearing trainers OR shorts.'	1 hour	Unit 1 LO1, LO2 Unit 15 LO3
Strings and loops – Hangman part 1	String manipulation can be taught with the build-up of a hangman game. Initially a program can be class-taught which takes a word from the keyboard and shows the correct number of blanks on the screen. This could be used to introduce fixed loops to iterate the length of the word. Version 2 of the program might take a letter input and show where that letter falls within the word. Depending on the language being used it is likely that built-in string manipulation functions can be introduced here. Version 3 might add a check to determine if all of the word has been guessed and a maximum number of wrong guesses. Problems along the way can be set for the learners who are progressing quickly, such as 'How might you prevent an identical guess being made?' or 'How would you deal with words containing repeated letters such as 'hello' and 'success'?'.	1 hour	Unit 1 LO1, LO2 Unit 15 LO3

Title of suggested activity	Suggested activities	Suggested timings	Also related to
Encapsulation – Hangman part 2	As the program builds up, separating additional parts into separate procedures (or methods) with meaningful names such as 'CheckForEndOfGame' or 'ShowHangManOnScreen' will inherently teach encapsulation before even discussing it with learners. By the end of creating this game, learners will have built up examples of most of the programming constructs listed within the Teaching Content. An extension can be to provide a series of simple graphics as the hangman visual feedback rather than a count of wrong guesses. Other extensions could be to provide a two-player option where the word is typed in, or a single-player option where the word is randomly selected from a bank of pre-programmed words.	1 hour	Unit 1 LO1, LO2 Unit 15 LO3
Maths game	Learners are gaining confidence by now but will most likely lack independence in researching solutions to problems on their own. This game focuses on answering a series of simple arithmetic questions against the clock. The program can produce at random questions such as '6 + 9' and the player must answer correctly either by clicking on the correct answer or typing it in. Results can be shown at the end, such as how long 20 questions took and how many correct answers were given. Writing this program will require the use of several variables to keep track of the current question, the time and the current score.	1 hour	Unit 1 LO1, LO2 Unit 15 LO3
Calculator	Learners can provide a simple addition calculator. Extensions could be providing subtraction, the C and CE buttons, etc. This project can be surprisingly complex and has the capacity for extensions into the realms of a scientific calculator (example scientific calculator: http://www.calculator.net/scientific-calculator.html).	1 hour	Unit 1 LO1, LO2 Unit 15 LO3

Title of suggested activity	Suggested activities	Suggested timings	Also related to

Times table production	This simple program could be used to teach fixed loops to produce a times table on the screen. For example:
	1 × 9 = 9
	2 × 9 = 18
	
	In Microsoft Visual Basic.Net, the code could be:
	`    Dim iCount As Integer`
	`    Dim intResult As Integer`
	`    For iCount = 1 To 12`
	`      intResult = iCount * 9`
	`      MsgBox(CStr(iCount) + " x 9 = " + CStr(intResult))`
	`    Next`
	
	In this example, note the use of the built-in function "CStr" which converts from the data type integer to data type string in order to display the result in a string message box.
	
	An extension could be to input which times table to create.
	
	In Microsoft Visual Basic.Net, the code could be:
	`    Dim iCount As Integer`
	`    Dim intResult As Integer`
	`    Dim intNumber As Integer = InputBox("Please enter a number")`
	`    For iCount = 1 To 12`
	`      intResult = iCount * intNumber`
	`      MsgBox(CStr(iCount) + " x " + CStr(intNumber) + " = " +`
	`CStr(intResult))`
	`    Next`
	
	Further extensions could be to place the results in a listbox and to format the output numbers such that the units line up correctly; for example:
	1 × 9 = 9 (The 9 lines up with the unit column)
	2 × 9 = 18
	
	An introduction to visual controls for Microsoft Visual Basic.Net can be found here:
	http://www.tutorialspoint.com/vb.net/vb.net_listbox.htm.
		1 hour	Unit 1 LO1, LO2
			Unit 15 LO3

Title of suggested activity	Suggested activities	Suggested timings	Also related to
Input verifier	A very simple program to ask the learner to input their age. The program should be made robust against incorrect data entry of different types: • Data presence: No data entered. • Data type: e.g. 'aaa' is entered as an age. • Data range: negative ages or an age over 130 years. This program could be used to teach conditional loops as the program will not know how many times to repeat the input (Bitesize introduction to fixed and conditional loops: http://www.bbc.co.uk/education/guides/zy38d2p/revision/10).	1 hour	Unit 1 LO1, LO2 Unit 15 LO3

SUGGESTED ACTIVITIES

LO No:	**2**		
LO Title:	**Be able to investigate business requirements for programming solutions**		
Title of suggested activity	**Suggested activities**	**Suggested timings**	**Also related to**
Kate's Pizza	Many small companies require a simple program tailored to their specific way of doing things. This provides an excellent opportunity for a local business owner or an instructor to pose as a local business owner to propose software needs to the learners. An example scenario is as follows: Kate runs a small pizza delivery company. She currently keeps paper records which start as written phone orders and end up being a pile of order notes which then have to be collated by hand to calculate the day's takings. She requires a computer application to help her run her business. Orders are taken over the phone and hand written onto a pad of A6 sheets. Payment is taken immediately over the phone via credit or debit card or as cash on delivery. Payment method and anything else required to make and deliver the order is written on the A6 sheet. Once the call is complete, the sheet is ripped from the pad of blanks and passed to the kitchen on a rotary queue. The cook processes orders on a first come, first served basis. Once the order is ready, the order sheet is placed on top of the completed order ready for the delivery department to pick it up. As delivery drivers come available, they look at the completed orders and pick a few that they know are in the same area. This sounds chaotic but it actually works well, as they have enough delivery personnel with sufficiently good local route knowledge to plan efficient round routes in their heads. When a delivery driver returns after completing deliveries, he or she returns any payments taken to Kate and sticks the completed delivery notes onto a spike ready for collating at the end of the day. Kate, the owner-manager, takes this spike home at close of business to calculate the day's takings. She adds up each order with a calculator. It is a busy business. At peak times, the handwritten orders are barely legible. For the most part, orders are completed to the customer's satisfaction, but a few order notes are inevitably lost before reaching the big pile of orders on the spike. Kate wants a computer system to help the existing system run more smoothly. If possible, she would like to work less in the evening. Ideally, she would like tools for analysing the data to spot best-selling pizzas and to keep an eye on her delivery personnel.	3 hours	Unit 1 LO3 Unit 2 LO3, LO4 Unit 6 LO2 Unit 9 LO2 Unit 11 LO2 Unit 12 LO3 Unit 21 LO2

Title of suggested activity	Suggested activities	Suggested timings	Also related to
Kate's Pizza (continued)	From this scenario, requirements can be drawn such as: 1. A quick way of filling in phone orders. 2. Be able to order a custom or preset pizza. 3. Time and date each order. 4. Record each order in a file for calculating the day's takings. 5. Print each order to keep the kitchen system the same. 6. A way of assigning a driver to a delivery. 7. Daily report print capabilities including money totals. 8. A way of updating the menu and prices. In requirement 1, the word 'quick' can be used to demonstrate ambiguous requirements, as 'quick' to a geologist might be 500 years! Learners should realise that the most complex coding solution is not always the best. Requirement 6 might be simply providing a box on the printout in which the driver writes his name. Requirement 7 can be done with a spreadsheet if the program saves each order to a text file that the spreadsheet can load. Requirement 8 could be used as an extension for the more advanced learners. Kate's pizza menu can be simulated so the learners have a fixed product list and prices to work with or they could use online companies for reference. For the solution to be merit rather than pass, it has to address the majority of questions that a software engineer would have if given the document and tasked with producing the system. It is not expected to be perfect but it should provide the basis on which a program can be designed. To achieve D1, learners can discuss which requirements are critical and which are not. They can also determine which might go into the first release to demonstrate to the client for initial feedback and which requirements will wait for subsequent prototypes. Learners can discuss the problem of integration with existing systems, which in the example scenario means the kitchen and the delivery system. They could discuss the feasibility of a small business being able to afford a system that integrates sat-nav combining nearby orders for the same driver or an online ordering system for customers to submit their own orders. They could investigate the time and cost of integrating the post-code address database from the Royal Mail.		

Title of suggested activity	Suggested activities	Suggested timings	Also related to
Petrol pumps	Design a program to simulate the process of clearing the display, allowing stop-start of fuel flow, calculating price from flow, finishing the transaction and paying so as to be ready for the next customer. This project initially feels simple but sufficient complexity is provided by the stages which a fuel transaction must complete and the uncertainty of the customer stopping and continuing the flow. Extensions could be different fuel types and costs and the ability to read out totals of fuel pumped and money taken over a set period of time. Suggested inputs: Fuel selection from a number of options (e.g. unleaded, premium, diesel) Flow start button Flow stop button Transaction complete button. Suggested processing: Disallow fuel selection once delivery commences Running total of litres delivered × price per litre of the fuel selected Flow stop allows flow start for the transaction to continue Transaction complete zeros the transaction for the next customer. Suggested outputs: Litres delivered Price per litre of the fuel selected Transaction price. Extension ideas: Calculate the proportion of the transaction that was tax. Provide a supervisor with total transaction summary, including a total of each fuel type and an overall transaction total.	3 hours	Unit 1 LO3 Unit 2 LO3 Unit 6 LO2 Unit 9 LO2 Unit 11 LO2 Unit 12 LO3 Unit 21 LO2
Stock control	A grocery store has a number of products and differing stock levels for each product type. Design a program to keep track of purchases and deliveries of stock. A fictitious inventory should be provided to the learners so definite requirements can be extracted. Extensions could be: some products have a finite shelf life; the ability to cope with spoiled goods and missing goods.	3 hours	Unit 1 LO3 Unit 2 LO3 Unit 6 LO2 Unit 9 LO2 Unit 11 LO2 Unit 12 LO3 Unit 21 LO2

Title of suggested activity	Suggested activities	Suggested timings	Also related to
Prioritise requirements	For D1, learners can discuss which requirements are critical and which are not. They can also determine which might go into the first release to demonstrate to the client for initial feedback and which requirements will wait for subsequent prototypes. Learners can discuss the problem of integration with existing systems, which in the Pizza example scenario means the kitchen and the delivery system. They could discuss the feasibility of a small business being able to afford a system that integrates sat-nav combining nearby orders for the same driver or an online ordering system for customers to submit their own orders. They could investigate the time and cost of integrating the post-code address database from the Royal Mail. Those being distinction criteria, this activity can be set as an extension for those learners able to complete the design before others.	1 hour	Unit 1 LO3 Unit 2 LO3 Unit 6 LO2 Unit 9 LO2 Unit 11 LO2 Unit 12 LO3 Unit 21 LO2
Requirements capture	With the customer present, or a person posing as the customer, learners could ask questions about the current system and what needs to be changed. Questionnaires and other fact-finding methods could be used and later submitted as evidence. Learners could be tasked with identifying local businesses to politely inquire about their systems, both manual and automated.	1 hour	Unit 1 LO3 Unit 2 LO3 Unit 6 LO2 Unit 9 LO2 Unit 11 LO2 Unit 12 LO3 Unit 21 LO2
Finished document	For the solution to be merit rather than pass, it has to address the majority of questions that a software engineer would have if given the document and tasked with producing the system. It is not expected to be perfect but it should provide the basis on which a program can be designed. Learners should collate all of their design activities and produce a discrete document complete with a table of contents (TOC) and executive summary.	1 hour	Unit 1 LO3 Unit 2 LO3 Unit 6 LO2 Unit 9 LO2 Unit 11 LO2 Unit 12 LO3 Unit 21 LO2

SUGGESTED ACTIVITIES

LO No:	**3**		
LO Title:	**Be able to develop software solutions to meet business requirements**		
Title of suggested activity	**Suggested activities**	**Suggested timings**	**Also related to**
Designing for a software engineer that isn't you	Learners will need to create a design specification that may build on LO2 that can be handed to a programmer so they can create the program with minimal further questioning. The specification will show what the graphical user interface (GUI: http://searchwindevelopment.techtarget.com/definition/GUI) will look like and what the program will do. The GUI design is likely to evolve as the project progresses; the differences between the end product and the initial designs can contribute to the M2 adaptations evaluation. It is acceptable to use a GUI designer provided by an IDE to design the GUI, such as Microsoft Studio, regardless of whether the same language is used for creating the program. It would be short-sighted to dismiss a GUI design tool just because it is linked to a language where no code is required at the design stage. Alternatives to using a GUI design tool are word processing drawing tools or drawing by hand onto paper. Drawing by hand lends itself to group discussion activities where learners can collaborate as long as they all contribute. Discussions can compare the speed of hand drawing against the ability to edit designs using a software tool, which will develop their design skills.	3 hours	Unit 6 LO1 Unit 11 LO3 Unit 15 LO2
Adapt to enhance	Adaptations are part of the M2 evaluation. This could be after a prototype has been demonstrated to the customer or as feedback to the design document. The customer could provide several points about feedback to which the learner responds with potential changes to the prototype. The changes could be different products that were not originally listed in the requirements, as the business world rarely stands still.	1 hour	Unit 1 LO4 Unit 15 LO2, LO3
Third party code	The learners could be provided with a small amount of code which solves a specific problem and asked to integrate it with their own project. This can be part of their final program as long as it is properly referenced as third party code. For example, a module or class which saves a given order to the end of a text file could be provided to learners struggling with requirement 7 in the pizza example scenario. This reflects how a software engineer would reuse other people's code to solve a problem that has already been solved elsewhere.	1 hour	Unit 15 LO3
Coding	Learners should be given a significant amount of time in which to develop their prototypes. During this time, individual learners will encounter different problems and the teacher will respond accordingly. It is crucial that the teacher is familiar with the programming environment chosen to develop the prototype and it is strongly recommended that all learners use the same environment that they have been using so far in the unit.	6–10 hours	Unit 15 LO3

Title of suggested activity	Suggested activities	Suggested timings	Also related to
Would you do it differently if you did it again?	For D2 learners must evaluate how each requirement was met or attempted. Learners should use their M1, M2, D1 and ideally M3 to feed into D2. For each requirement, learners will note how successful their solution was in satisfying the requirement and how an alternative approach might have been a better solution. In other words, for D2, learners can take each requirement in turn and evaluate how well their approach solved the requirement and also discuss possible alternatives that could have been implemented. Many software engineers recognise that the benefit of hindsight allows them to see better solutions than the one they chose. A learner can describe success or inadequacies, as it is the analysis in the justification that is being assessed. Learners could keep a development diary or annotate their code to later support this activity.	2 hours	Unit 8 LO4
Testing, testing	Learners should create a test table which describes a series of activities which tests each requirement of their program. Learners are likely to leave out critical information in the test plan as the learners themselves are very familiar with their own projects. To demonstrate this, the tests could be carried out by other learners, so the test plan is used by a classmate. The tests to be carried out may have been identified at the design stage to support the desired outcomes. When running the tests, it will become apparent what information is missing from the test plan to make the tests repeatable. For example, in the pizza scenario, a test might say 'order a preset pizza and check the price is correct' but not say which pizza and how much the total should be. Feedback can then be used to enhance the test plans. 1. Create the test plan. 2. Use the test plan and gather feedback. 3. Enhance the test plan and retest.	3 hours	Unit 9 LO3 Unit 15 LO3 Unit 21 LO3

SUGGESTED ACTIVITIES

LO No:	**4**		
LO Title:	**Be able to propose software solutions to meet business requirements**		
Title of suggested activity	**Suggested activities**	**Suggested timings**	**Also related to**
I have to do WHAT?	Learners have varying levels of confidence and competence with public speaking (guidance for beginning public speaking: http://www.gingerpublicspeaking.com/how-to-start-a-speech/); (wordy guidelines for public speaking from the University of Florida: https://edis.ifas.ufl.edu/wc023). Common mistakes well known to instructors include speaking to the wall rather than the audience, mumbling, rushing material, poorly auto-timed slides, too much written content, failing to introduce themselves properly, failing to give a definite ending or simply being paralysed with fear. The fact remains that industry needs engineers with communication skills and these skills need to be acquired with practice. It is recommended that opportunities to practise start from the first lesson. Learners can be asked to introduce themselves to their neighbours in the class then spend 10 seconds introducing their neighbours to the whole class. One learner a week could be asked to introduce the day's lesson to the rest of the class, or read the day's headlines, or the weather report.	5 minutes at the start of each lesson	Unit 1 LO4 Unit 6 LO4 Unit 7 LO4 Unit 10 LO4 Unit 15 LO4 Unit 22 LO3
How bad can it be?	The tutor or learners can prepare and give a presentation that is deliberately poor. Some examples available online are: Bridget Jones introducing a book launch event: https://www.youtube.com/watch?v=tclaT5D4Iwc Short video created by young learners on the "Seriously Extreme Overuse of the Filler-word Like". https://www.youtube.com/watch?v=ol1BN5kHj0U Judge Judy exasperated at a youth's inability to form a sentence without using the word "like": https://www.youtube.com/watch?v=yyPMbjoDSJY Compilation of bad public speaking habits demonstrated by learners: https://www.youtube.com/watch?v=92Yz5r3Jc6k Humorous sketch demonstrating poor presentation techniques: https://www.youtube.com/watch?v=69JZD60eR6s A presentation given poorly with mistakes listed at the end: https://www.youtube.com/watch?v=ATfY8dvbuFg The same presentation given better: https://www.youtube.com/watch?v=5utoLhjUuAI	2 hours	Unit 1 LO4 Unit 6 LO4 Unit 7 LO4 Unit 10 LO4 Unit 15 LO4 Unit 22 LO3

Title of suggested activity	Suggested activities	Suggested timings	Also related to
How bad can it be? (continued)	In pairs or individually learners can be given a specific mistake to demonstrate, such as speaking to the wall or mumbling, and the audience must guess which trait is being demonstrated. These presentations would last only two minutes or so and the content could be trivial such as popular pets or favourite football team. Animated entertaining short clip giving presentation tips: https://www.youtube.com/watch?v=tShavGuo0_E A good pitch to the Dragons' Den: https://www.youtube.com/watch?v=zumkSeC2u3M		
Present to different audiences	Learners divided into teams could be given another micro-presentation to prepare for; this time they are all the same topic but the audience identified will be different. For example, talking to the marketing department about who might buy your software is a different presentation from talking to the software engineers who might maintain the software product.	2 hours	Unit 1 LO4 Unit 6 LO4 Unit 7 LO4 Unit 10 LO4 Unit 15 LO4 Unit 22 LO3
Practise, practise, practise	The more practice learners get in public speaking, the more relaxed they will be with public speaking. Positive feedback is important to build confidence; an early class starter might be to grade a list of feedback comments as positive or negative and constructive or destructive, comparing comments such as: 'I don't like the shade of blue you used' against 'A lighter blue would show up better against your background'. Learners could then be trusted to give positive and constructive comments to each other. A mock micro-presentation can be designed to give learners more experience in front of an audience. Each learner can be given 30 seconds to introduce themselves and something unique about their project: for example 'Hello, my name is Arran and I have chosen to use multiple forms instead of one because it splits the process into easier steps.' Prompt cards and scripts could be banned to promote increased audience eye contact. Learners could end with 'Any questions?'. To promote good practice positive and constructive feedback could be given by the learners themselves. Care should be taken that learners listen to the presenters rather than be distracted by working on their own presentation. The format could be quick-fire presentations to get round the class in one lesson, or prepare in one lesson and present in the next.	2 hours	Unit 1 LO4 Unit 6 LO4 Unit 7 LO4 Unit 10 LO4 Unit 15 LO4 Unit 22 LO3

Title of suggested activity	Suggested activities	Suggested timings	Also related to
Smile, you're on camera	A date should be fixed in advance for all learners to present five-minute presentations to their stakeholder. The stakeholder can be another teacher or a willing visitor; the unfamiliar face will add to the formal nature of the day. Senior management sometimes enjoy being the audience on such occasions and it is a PR opportunity. This is also an opportunity to engage a local employer to fulfil the meaningful employer requirement of the Technicals qualification family. Note that P4 is for learners to give a demonstration of their project; the quality of the presentation is not graded as long as it can be understood. A smart-phone video would be sufficient evidence of the activity. Feedback could be given at the end of the presentation for the project enhancement. Video evidence is the easiest form to use to prove to a moderator that the presentations took place. Modern smartphones are capable of capturing video of sufficient quality to recognise learners giving a presentation. In the rare case that a learner is incapable of delivering a presentation on medical grounds, then that learner could submit a video presentation to the panel as an alternative.	1–2 hours	Unit 1 LO4 Unit 6 LO4 Unit 7 LO4 Unit 10 LO4 Unit 15 LO4 Unit 22 LO3
Understand then analyse	Deliberately vague feedback could be suggested to the learner for enhancing their project, such as 'I don't like the colours'. The learner could ask some questions to a patrolling tutor or as a whole class activity to clarify what is meant for the enhancement and then describe how their project would have to change to accommodate the enhancement. It could be established that 'I don't like the colours' really means there is a company colour scheme that should be used or that the stakeholder is colour blind. This leads directly into M3 for enhancing their project due to feedback.	2 hours	Unit 1 LO4 Unit 6 LO4 Unit 7 LO4 Unit 10 LO4 Unit 15 LO4 Unit 22 LO3
Maintenance – creating version 2	Code the changes resulting from the activity above into their projects. Be sure that the learners document the changes as the adaptations have to be evidenced for M3.	2–4 hours	Unit 1 LO4 Unit 6 LO4 Unit 7 LO4 Unit 10 LO4 Unit 15 LO4 Unit 22 LO3

The small print

We'd like to know your view on the resources we produce. By clicking on the 'Like' or 'Dislike' button you can help us to ensure that our resources work for you. When the email template pops up please add additional comments if you wish and then just click 'Send'. Thank you.

Whether you already offer OCR qualifications, are new to OCR, or are considering switching from your current provider/awarding organisation, you can request more information by completing the Expression of Interest form which can be found here: www.ocr.org.uk/expression-of-interest

Looking for a resource?

There is now a quick and easy search tool to help find **free** resources for your qualification:

www.ocr.org.uk/i-want-to/find-resources/

ocr.org.uk/it

OCR Customer Contact Centre

General qualifications

Telephone 01223 553998
Facsimile 01223 552627
Email general.qualifications@ocr.org.uk

OCR is part of Cambridge Assessment, a department of the University of Cambridge. *For staff training purposes and as part of our quality assurance programme your call may be recorded or monitored.*

Cambridge Assessment

LLOYD'S REGISTER · LRQA ISO 9001 UKAS MANAGEMENT SYSTEMS 001