7 Development

Briefing

The common theme throughout this unit is the development of software and websites. Students will talk about determining the scope and specifications of a project (requirements analysis), the structure of websites, programming and how projects are managed.

Students practise: language to express user requirements; the passive, in the context of a web developer's work; **make** and **cause** to explain how something controls something else; the language of schedules.

If you are unfamiliar with programming, it is recommended that you check through the Briefing notes below and the audio script for the third section (Software development) carefully before the lesson. Rest assured, though, that the topic is dealt with at a very basic level.

Requirements analysis

This section begins by looking at the process of writing software and websites to meet client needs but the main focus is the initial stages of this process: working out the requirements and producing a set of specifications, a process often known as **requirements analysis**. The rest of the unit will look at the other parts of the process.

The person who is responsible for the requirements analysis process is the **systems analyst**. The process that the software or website needs to follow has to be documented in minute detail so that the programmers know exactly what to do. It needs to cater for every eventuality that might occur. An early step is to identify the stakeholders, which is anyone who will directly or indirectly be affected by the software. This could include, for example, customers of the client who will use the software, as well as employees of the client. In most cases these people are then interviewed (see the interview on page 52 of the Course Book for more information.)

The result of the process is a specifications document which the client is generally asked to check and agree to before the programming commences.

Website design and architecture

Several aspects of website architecture are common to most websites: most have menus near the top of each page, copyright and links to legal disclaimers at the bottom, links to other pages down the left or right side and so on. Also, similar menu items appear in many websites: there are usually items for contact details, 'about us' (which gives information about the organisation), **FAQs** (Frequently Asked Questions) and 'contact us' or 'contact details'. Text in websites is often called **hypertext**, that is, text containing links (**hyperlinks**) which can be clicked on to go to other pages within the site or external websites.

Website development can involve a range of programming technologies. The most basic of these is **HTML** (Hypertext Markup Language). Version 5 of this (**HTML5**) includes provision for video and animations; older pre-HTML5 websites had to use other technologies such as Flash to display moving images. Many websites use databases to store such things as user login details; one that is commonly used is **MySQL** (pronounced either 'my S, Q, L' or 'my sequel'). Many websites generate pages 'on the fly', that is, rather than being pre-written, pages are created in response to user input. These are called **dynamic** web pages. Facebook pages are a good example: every time someone accesses their Facebook page, it may appear different because it shows content from other people's pages that may have been updated since the last visit. Generation of these is often carried out by a kind of programming language called a scripting language that is embedded in the HTML of the web page; examples include **PHP** (open source) and **ASP** (a Microsoft product).

Website designers often have to pay attention to **search engine optimisation** (**SEO**); that is, they have to design the site and the text within it such that it is seen as an important website by the major search engines such as Google, and thus appears near the top of search results (i.e. has a high **search ranking**).

Software development

A large number of programming languages exist. Some common ones are Java (often used for web applications) and the **C** family of languages, which includes C, C++ and C# (C sharp). Students may have been exposed to languages such as BASIC at school. The examples used in this unit are from C.

An important concept in any programming language is the **variable**. Variables are represented by names that are one or more characters long and they have a value which can be a number, a character or a string of characters. As their name suggests, their value can vary; this occurs in response to instructions (otherwise known as commands) within the program. Two examples of short sections of programs (or **source code**, often just called **code**) are provided on page 56 of the Course Book.

The first gives the value 3 to a variable called 'a' in the first line, then the value 2 to another variable called 'b' in line 02. It then adds them together, putting the result of this sum into another variable, 'x'. The final instruction outputs the value of x (5) to the screen.

The second section of code is part of a program that allows a robot to be controlled from a mobile phone: the key that is pressed on the phone determines the direction of the robot's movement. The code begins by setting two variables, 'g_Move' and 'g_Turn', to zero. After that, another variable called 'key_Press' comes into play; this takes the value of the key that is pressed on the mobile phone (to avoid complication, the exact means through which this is done is not discussed). Then there are four lines beginning with 'if'; these all work in a similar way to each other. The first one checks the value of 'key_Press'; if the value is 'a', i.e. if the 'a' key has been pressed on the mobile phone, then the value of 'g_Move' is changed to 1. As we hear on the audio, this means that the robot moves forward (how it does this would be dealt with later in the program, and again, for simplicity, is not discussed in the unit). The next line sets 'g_Move' to 2, corresponding to the robot moving backwards, and so on.

Project management

Software development projects are notorious for taking much longer and being more expensive than originally anticipated. For this reason, project management within programming is very important. Most project management software produces Gantt charts, which generally show time across the horizontal axis and parts of the project down the side. Thus, people involved in the project can see at a glance where each aspect of the project should be at any particular time and how a delay in any part of the project could affect other aspects of the project.

A typical software development project has several stages. After the requirements analysis (see the first section of this unit), the programmers do the **coding**, that is, writing the software. When they have a working prototype of the application, **alpha testing** commences. This is usually done internally, without client involvement, and focuses on the basic functionality of the software. Any major bugs identified are then ironed out by the programming team and the software then goes to **beta testing**. This generally involves end users using the software and reporting on any problems they find. After a further round of debugging and perhaps further testing, a **release candidate** is produced. If there are no significant problems with this, the release candidate becomes the final version.

Business matters

In this section students read a scenario involving a website development project. They make decisions about the content of the website, plan the project using a Gantt chart and write a report about it.

Further reading

Use the following keywords to search the internet for websites which give more in-depth information about the topics covered in this unit: systems analyst, requirements analysis, web design, HTML, Adobe Flash, MySQL, dynamic web page, PHP, programming language, C (programming language), software development, software testing.

Teacher's Notes

Warm-up

Ask students to share the homework they did at the end of the previous unit: the stages in writing a program and/or the common features of websites that they looked at.

Requirements analysis

1 If your students are familiar with software development, discuss with them the stages in the process, writing them on the board as you go, or ask them to come up with a sequence of stages in pairs. Otherwise, ask them to work in pairs, open their books and put the stages provided in order. Some vocabulary appears here for the first time. It is dealt with in Activity 4; students should be able to do Activities 1, 2 and 3 without it.

5 The customer checks and approves the final version.
1 Speak with the people who will use the new software and analyse how they will use it.
2 Plan the project, write the specifications and prepare instructions for the programmers.
4 Test and debug the code.
3 Write the code.

Reading

2 The work of a systems analyst is explained through an online magazine interview. Students skim it to decide which of the stages from Activity 1 are mentioned.

She mentions 1, 2 and 3.

3 Students now answer some more detailed questions about the interview in Activity 2.

1 to find out who the users will be
2 They have to look at every step in the process carefully and in a lot of detail.
3 flow charts and drawings of the user interface
4 to be sure they are happy with it

Vocabulary

4 Students match words from the interview to their definitions.

After the feedback stage, ask students to guess what *bug* means from the meaning of *debug*. With stronger students, it would be helpful to go through (or elicit from them) any other parts of speech of these words: for example, the noun form of *analyse* is *analysis*.

1 detail **2** analyse **3** approve **4** interview **5** debug

Extra activity

Ask students to close their books. Write the vocabulary from Activity 4 on the board. Then put students in pairs and ask them to explain the software development stages to their partner using the vocabulary on the board. This activity is a good way to review and recycle vocabulary and can be applied to many situations throughout this book.

Listening

5 ▶ 36 This and the next three activities involve a scenario in which an IT company is developing a website for a pizza shop to enable its customers to order pizza online. If your students are not from a part of the world where pizza shops exist, it will help to show them photos of pizzas and pizza shops; there may be local food that is similar. Make sure that they know the words *topping* and *base* and that pizza shops usually offer a few standard toppings such as Margherita (a basic one with cheese and tomato); also, that customers can often choose their own combinations of ingredients as a topping.

Students will hear a systems analyst for the IT company talking to a worker. At this point, they listen for gist: which stage in the software development process (see Activity 1) is he at?

He is at the interview stage.

6 Draw students' attention to the flow chart, which the systems analyst will create during the requirements analysis process. Note that it has been significantly simplified to avoid too much complexity in the lesson – in real life, a flow chart such as this would be much more complex. (for example, there is no mention of payment).

In pairs, students try to match 1–4 in the flow chart with the steps in the box before listening again; logic will help. Then they listen again and check their answers.

1 Customer wants standard pizza?
2 Ask which toppings.
3 Ask which type of standard pizza.
4 Write order on order sheet.

7 ▶ 37 Students listen to the next track and complete the rest of the flow chart. As with the previous activity, they can try to do this in pairs before listening.

5 Customer wants another pizza?
6 Ask for delivery address.
7 Calculate delivery time.
8 Tell customer delivery time.

Extra activity

Ask students, in pairs or small groups, to discuss which shapes of boxes in the flow chart represent the following:
1 *yes*/*no* questions
2 input/output
3 thinking (for humans) or processing (for computers)

Answers:
1 the diamond-shaped boxes with *Yes* and *No* arrows coming from them
2 the rhomboids
3 the rectangles

Language

The Language box brings together some language points that can be used to express user requirements. Students will be familiar with their form but perhaps not with using them for this function. To elicit the language in the top row, you could ask them to tell you what a pizza shop's website should do as well as take orders (for example, it should show pictures of the pizzas available). By asking students to express these from the shop owner's point of view, the language in the bottom row of the Language box can also be elicited.

Speaking

8 In pairs, students use the flow chart in Activity 6 and language from the Language box to express the requirements for the pizza shop website.

9 The situation switches from a pizza outlet to a different kind of shop: a clothes shop. Using language from the Language box, students work in small groups (or pairs) to list requirements for their online ordering system.

10 Students compare the list they made in Activity 9 with another group's list. In addition, you could ask them to discuss any differences between their respective lists, then compile a new list bringing together the agreed most important requirements from both groups.

Website design and architecture

Speaking

1 Students will have seen many websites; this initial activity builds on that familiarity by asking them to discuss in pairs some general questions about websites.

Suggested answers
1 contact details, about us, home (page), FAQs
2 menus, title, links
3 easy to use: clear menus, not too much text, important things such as a search box in easy-to-find places, etc.; interesting: photos, videos, things to do (not just read)

Vocabulary

2 Students match common website menu items to a typical website. Make sure they understand that they will be putting the menu items under the most appropriate heading. There is not necessarily only one possible solution – any arrangement that can be justified should be fine.

1 FAQs **2** How to play **3** Images
4 Videos **5** How to pay **6** Prices
7 Company blog **8** Contact us **9** Players' forum **10** Login
Items in any particular column are interchangeable. Other answers may be possible as long as they can be justified.

Reading

3 Students read a case study about a website development project and answer the gist questions. There is some technical information within the text, such as the names of various technologies frequently used in website development (*PHP*, *HTML5*, etc.). Make sure students are aware that they do not need to know what all of these are to understand the text. They need to get used to the fact that many of the documents they will be reading in the workplace will contain a lot of these.

Suggested answers
A fan site is a site for people who play a particular game/for fans of a game.

Yes, the project was successful: we know because the customer was very pleased with the end result (final paragraph).

Vocabulary

4 Using context within the case study in Activity 3, students find vocabulary that matches the definitions provided.

1 public **2** content **3** premium
4 combination **5** versatile **6** challenges
7 viewable **8** search rankings

Language

One way to introduce the passive is to find an example in the case study and ask students to say who carried out the action and whether that information is stated in the sentence (for example: *PHP was chosen to keep costs down*. The developers carried out the action (chose PHP) but they are not mentioned in the sentence.).

Passive forms of the present and past simple are dealt with here, along with one modal verb: *can*. Check that students are aware of the form of the passive: *be* (in the appropriate tense) + past participle. Where *can* is used, it goes before *be*.

With stronger students, you could also bring in the point that the 'doer' of the action (often called the agent) can, if appropriate, be mentioned by placing it after *by* later in the sentence (for example: *A website was developed by Andrea*.). However, this is not necessary for the activities in the book.

Extra activity

Ask students to identify examples of the passive in the case study in Activity 3 that have not already been mentioned. This could be done in pairs. Students could also be asked to decide who carried out the action.

5 For controlled practice, students rewrite some active sentences in the passive. These could be checked in pairs.

1 A problem was found.
2 A dynamic, exciting website is required.
3 PHP was used for this website.
4 Videos can be watched on this website.
5 Useful PDFs can be downloaded from this website.

Speaking

6 Here students are introduced to a website navigation chart: a diagram that shows how pages in a website are linked. Some terminology for the different levels derives from the chart's resemblance to a family tree; point this out to students and elicit from them an example of a parent node, a child node and a grandchild node from the diagram. Then put them in pairs to continue the activity.

'News' and 'About SL8' are linked to the home page.
'Home' is the parent node.
'News' and 'About SL8' are the child nodes.
'History of SL8', 'FAQ' and 'Join the club' are the grandchild nodes.

7 In this information gap activity, each student in a pair has half of a website navigation chart. They describe what they can see to each other and complete their chart using the information they hear, without looking at their partner's chart. At the end, they can check their answers by looking at their partner's chart.

Writing

8 Students see the beginning of an email summarising what was done to build the website in the case study in Activity 3. They complete it after re-reading the case study, using the passive.

Suggested answers
- PHP was chosen for its low cost.
- HTML5 was chosen because it works with many systems.
- MySQL was chosen for the private area.
- The site was made viewable on mobile phones (and smaller tablets).
- SEO was worked on.

Software development

Speaking

1 The introductory questions for this section allow those with little experience of programming to begin to talk about the topic. Those with more experience could discuss a different set of questions depending on experience, such as:
1 Which programming languages have you heard of?
2 What programming have you done before?
3 Which language(s) did you use?
4 Which programming language do you prefer? Why?

2 Here students look at a basic example of programming code and explanations of two key terms: *programming instruction* (which can also be called *command* or simply *instruction*) and *variable*. *Line of code* is also mentioned but this should be quite straightforward. Students read the explanations and, in pairs, answer the comprehension questions. If you think that your students already know *vary*, then pointing out the connection with *variable* will be useful.

Note that *cout* is pronounced as the letter *c* followed by *out* (/si:/ + *out*).

If students have difficulty understanding what is going on here, write this on the board:
a = *7*
b = *3*
x = *a* + *b*
Then ask: *What is x?* The answer is 10. Then adjust the values of *a* and *b* and get students to tell you what *x* becomes.

1 The constants are 3 and 2.
2 x becomes *5* (*a* is 3 and *b* is 2; *c* is *a* + *b*, i.e. *c* is 3 + 2, which is 5.

Listening

3 ▶ 38 Explain to students that the next section of code is part of a program that controls a robot from a mobile phone. Pressing different keys on the number pad makes the robot move forward, backward, or turn left or right.

Ask students to look at the code (which is in the C programming language). Tell them they do not need to understand it yet but they will hear two programmers talking about it and will have to identify features while listening. Elicit from students that *g_Turn*, *g_Move* and *key_Press* are variables.

While listening, students write numbers in the boxes below the code to indicate the order they hear these variables mentioned.

1 g_Move **2** g_Turn **3** key_Press

4 Students complete the sentences to indicate what happens when the variables have certain values, then listen again to check their answers.

This is a good place to review the zero conditional from Unit 3 if you feel your students need it.

1 doesn't move **2** doesn't turn **3** x

5 ▶ 39 Students listen to the rest of the conversation and identify the lines of code the speakers are now talking about.

They are talking about the four lines beginning with 'if'.

6 Students can either listen again and write down the keys that correspond to the robot's direction of travel as indicated in the diagrams (as per the instructions in the Course Book), or write the answers without listening and, if necessary, check their answers while listening. Make sure they understand that the arrow in 1 points forward and the one in 3 points backward.

1 a **2** d **3** f **4** s

Speaking

7 The purpose of this activity is to provide further reinforcement of what the code in Activity 3 does, so if your students understood the code readily, the activity can be skipped. Students simply write the appropriate values for the variables in the gaps in the flowchart.

1 a **2** f
This part of the flowchart represents the first two instructions beginning with 'if'.

Extra activity
Students complete the flowchart to represent all four lines of code with *if* instructions.

Language

Ask students what the 'f' key does. If they have encountered *make* and *cause* previously, it should not be difficult to guide them towards producing the target language, though you may

have to help them to choose the correct verb form. Otherwise, simply provide them with this. To help make the meaning clear, you could also point out the parallels with the zero conditional: the first example could also be written as: *If you press the 'f' key, the robot goes backwards.*

8 For controlled practice of the language point, students produce sentences to represent what the robot does in response to each key press.

'f' makes the robot go/causes the robot to go backward.
'd' makes the robot turn/causes the robot to turn right.
's' makes the robot turn/causes the robot to turn left.

Speaking

9 Students are presented with some further programming instructions and explanations of what they do. In pairs, they use *make* and *cause* to ask and answer questions about what the instructions do. Before they begin the activity, draw their attention to the example conversation at the bottom of the activity, which they can use as a model.

10 The prompts here include vocabulary from earlier in the book. Students say what each item does, using *make* and *cause*, thereby extending the language point to other contexts.

Suggested answers
1 The 'Maximise' button makes the window/causes the window to fill the screen.
2 A right click makes a pop-up menu/causes a pop-up menu to appear.
3 A double click on a file icon makes the file/causes the file to open.
4 The 'Off' switch makes the computer/causes the computer to switch off.
5 The 'Send' button makes an email/causes an email to go to the recipient.
6 The 'Save' button makes the program/causes the program to save the file.

Project management

Speaking

1 Students look at a Gantt chart and, in pairs or small groups, speculate about (or, if they are already familiar with the concept, describe) what it is used for. They should use the notes at the bottom of the chart to help them.

Suggested answer
People might use Gantt charts to plan projects or tell people when to start and finish things.

Vocabulary

2 Students match words from the Gantt chart to the definitions provided.

1 coding **2** milestone **3** alpha testing **4** beta testing **5** feedback **6** release candidate

Listening

3 ▶ 40 Tell students that they will hear a software developer (programmer) and a project manager discussing the project in the Gantt chart. Ask them to decide whether they are near the beginning or end of the project.

near the beginning

4 Ask students to look at the Gantt chart again. They will listen to the conversation a second time, identify the mistake in the chart and correct it. After that, they can discuss their answers in pairs and, if there is some uncertainty, listen again to check.

The programmers should have two weeks, not one week, after the beta testing (weeks 15 and 16)./The beta testing should last two weeks, not three./The beta testing should finish at the end of week 14, not week 15.

Language

Various ways of expressing future scheduled actions are given here. There are several ways to introduce the language. One is to discuss with students how the future is expressed in English, elicit as many forms as you can, then ask them to identify the future expressions in audio script 40 on page 79 and finally check what they found against the Language box. Note that this approach incorporates Activity 5.

Note that, in addition to the points in the Language box, in the audio script *should* is also used to talk about something that is expected to happen. Also, there is one example of a verb (*start*) used in the present simple to indicate a timetabled event. How much time you spend on these will depend on how strong your class is; with many classes it will be best for now to focus just on the language in the Language box.

5 Students look at audio script 40 on page 79 and underline all examples of the language from the Language box.

The systems analysts are finishing their tasks at the end of ...

... your team is scheduled to start coding ...

... the second milestone, being ready for alpha testing, is due in week 9.

... the alpha testing is due to finish at the end of ...

... you're scheduled to deal with the feedback from that ...

Pronunciation

6 ▶ 41 Students look at the audio script on page 79, listen to the two short conversations and mark the stressed words on the script. After feedback, they practise these conversations with a partner, using the appropriate sentence stress.

1
A: When are we due to finish?
B: Tomorrow, I think.
2
A: What's your schedule next week?
B: Well, I'm starting a new project on Monday!

Extra activity

Ask students how the meaning would change if different words were stressed in the conversations in Activity 6.

Answer
The stressed word generally carries a stronger meaning, so by changing the stress, you can change the emphasis.

Speaking

7 In pairs, students practise the language from the Language box by asking and answering questions about the schedule in the Gantt chart in Activity 1.

8 Focus now turns to real life for some personalised practice: in pairs, students ask and answer questions about any schedule they may have associated with their work and/or their study.

Business matters

In this section, students read a scenario involving a website development project. They then work together to make decisions about the content of the website, plan the project using a Gantt chart and write a report about it. Much of the language from this unit is recycled here.

Speaking

1 Students read the scenario and, in small groups, answer two focus questions.

1 It has a version of the game that people can play online/try out on the website.
2 designing/developing the special online feature (the game)

2 Students work together in the same groups to discuss and make decisions about appropriate content for the website. They also draw a navigation chart for the website.

3 In the same groups, students plan their project using a Gantt chart and the stages in the box.

4 Each group of students joins another group, and each explains their own group's decisions and plans and discusses the differences. To increase the amount of speaking practice, pairs could be formed comprising one member of each of the original groups.

Writing

5 Review the features of reports mentioned in Unit 6. Then have students write a short report to a manager describing their website design and project plan, using the headings provided. If they have time, they could include the site map and Gantt chart their group produced in Activity 3 in their reports. Writing could be done individually or in pairs. Feedback could involve peer review: students swap reports with another student and decide whether the stages in the report have been followed as specified.

Preparing for the next unit

In preparation for **Unit 8**, ask students to think of a story from their own experience about a problem with IT equipment. Where possible, it should be relevant to their study or work life outside English classes but, if necessary, an item of consumer equipment would be acceptable. They should be prepared to tell their story at the beginning of the next lesson.