Unit 4
Explaining methodologies, using technical illustrations, and assertion-evidence approach

Overview

The findings or results of your study may not appear valid if others do not understand how you obtained them. This unit will help you identify the relevant language features in describing your approach and methods for two types of studies common in computer science: 1) experimental (e.g., testing an algorithm), and 2) developmental (e.g., building a website). The use of technical illustrations will also be introduced with reference to their interactions with the underlying text and common pitfalls. You will also be introduced to the assertion-evidence approach in oral presentations. This has been designed specifically for technical and scientific presentations and is popular in the fields of science, medicine, and engineering.

Learning outcomes

By the end of this unit, you will be able to:

- identify the main components and language features in the methodology section
- report technical information using figures and graphics
- apply the assertion-evidence approach to an oral presentation of your FYP

(Warm-up) Brainstorm the main components of a methodology section

Work with a partner. Brainstorm and write as many as possible main components of an approach or methodology section.

Hint: Think of your project plan and presentation. Examples are data to collect, variables to monitor, materials and equipment to use, etc.

4.1 Explaining methodology

The methodology section provides details on exactly 'how' you approach and assess a problem, obtain your results, or generate a solution. The depth and quality of this section is very important because readers will turn to this section to assess whether the results and solutions obtained are valid or repeatable.

The aspects of methodology are many and varied dependent on the type of discipline and underlying nature of the project, such as experimental research (e.g., evaluating a computation technique) or software development (e.g., constructing a website). While the 'method' presented in experimental research focuses on the setup, samples, and conditions of the experimental design, the 'method' in software development is generally driven by user or functional requirements and can be better captured in separate sections represented by the design, implementation and testing in the software development life cycle such as overall system architecture design or database design. However, many generic components, such as the choice of equipment and variable selection apply in both types of projects.

Some components in presenting your methodology are suggested in Table 1 below. You should emphasize your 'considerations' and 'approach' in your report (more on justifying engineering solutions in Unit 5).

Experimental	Developmental (e.g., an online application)
• overview of the experiment setup • variables selection • sample (e.g., animate (e.g., human) or inanimate (e.g., machines) • sampling techniques • location • materials/equipment and setup • procedures • data treatment • restriction/limiting conditions • algorithm design (algorithm development)	• equipment/platform set up • system architecture design • user interface design • database design • user access/security design • test data and test case design

Table 1. Components in presenting methodology in experimental and developmental studies in the discipline of computer science

TASK 4.1 Identify the components in a methodology section

Below is a description of the method section of a report on evaluating the performance of *Blueprint* – a tool that integrates Web search into the development environment (the same report from which the introduction text was presented in Unit 3). A preview of the results is also shown to help you understand the method.

Step 1

Read the extract quickly. What type of research is it?

Step 2

Read the extract again and identify the components of the methodology section using the table from 4.1.

Hint: You may find components not listed in the table and some components in the table may not be present in the text.

Step 3

Answer the questions below:

1. Overall, is the method good enough to test the three hypotheses?

2. Is the sampling size big enough?

3. Looking at the results is the difference between using Blueprint and not (the control) significant enough?
   - Directed Task Completion Time –
   - Directed Task Code Quality –
   - Exploratory Task Chart Quality –

4. The participants had to complete the following:
   a) A tutorial
   b) A directed programming task
   c) An exploratory programming task

   Is there enough information about each one? Is there anything missing that you would like to know as a reader?

   a) A tutorial –
   b) A directed programming task –
   c) An exploratory programming task –

Text 1 [1]

EVALUATION

We conducted a laboratory study to better understand how Blueprint affects the example-centric development process. This study evaluated three hypotheses:

H1: Programmers using Blueprint will complete directed tasks more quickly than those who do not because they will find example code faster and bring it into their project sooner.

H2: Code produced by programmers using Blueprint will have the same quality as code written by example modification using traditional means.

H3: Programmers who use Blueprint produce better designs on an exploratory design task than those using a web browser for code search.

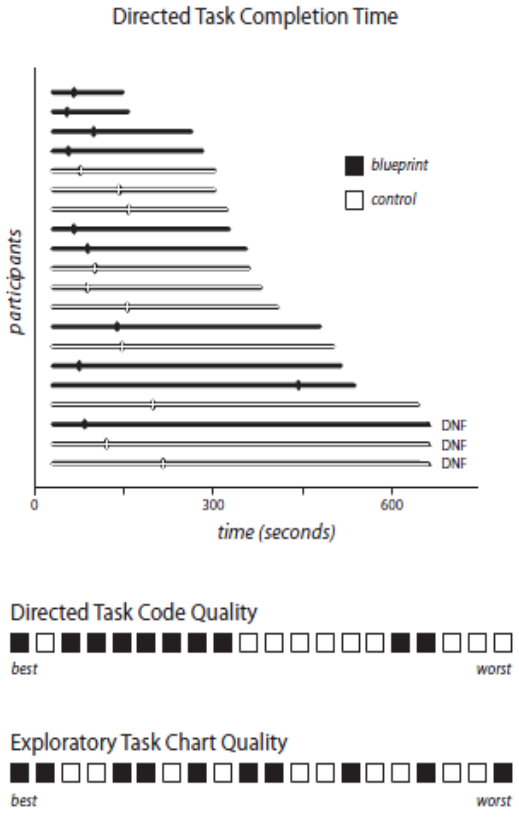


Figure 3. Results from the study. The top graph shows each participant's completion time on the directed task. Time of first paste is shown as a tick mark; participants who used Blueprint are shown in black. The two bottom graphs show ranking of participant's code quality (directed task) and chart quality (exploratory task), respectively. Again, participants who used Blueprint are shown in black.

Method

Twenty professional programmers (16 men, 4 women) participated in this study. We recruited them through an internal company mailing list and compensated them with a $15 gift card in exchange. Participants had an average of 11.3 years of professional experience. Fourteen reported at least one year of programming experience with Flex; twelve reported spending at least 25 hours a week programming in Flex.

Participants were given an off-the-shelf installation of Flex Builder, pre-loaded with three project files. Participants in the control condition were provided with the Firefox Web browser; they were asked to use the Adobe Community Help Search engine to look for example code. Participants in the treatment condition were provided with Blueprint to search for code samples; they were not allowed to use a web browser.

Participants were asked to complete a *tutorial*, a *directed task*, and an *exploratory task*. Participants were told that they would be timed and that they should approach all tasks as though they are prototyping and not writing production-level code. Participants began each task with a project file that included a running application, and they were asked to add additional functionality.

For the *tutorial* task, the sample application contained an HTML browsing component and three buttons that navigate the browser to three different Web sites. Participants received a written tutorial that guided them through adding fade effects to the buttons and adding a busy cursor. In the control condition, the participants were asked to use the web browser to find sample code for both modifications. The tutorial described which search result would be best to follow and which lines of code to add to the sample application. In the treatment condition, the participants were asked to use Blueprint to find code samples.

For the *directed programming* task, the participants were instructed to use the *URLLoader* class to retrieve text from a URL and place it in a text box. They were told that they should complete the task as quickly as possible. In addition, the participants were told that the person to complete the task fastest would receive an additional gift card as a prize. Participants were given 10 minutes to complete this task.

For the *exploratory programming* task, participants were instructed to use Flex Charting Components to visualize an array of provided data. The participants were instructed to make the possible best visualization. They were told that the results would be judged by an external designer and the best visualization would win an extra gift card. Participants were given 15 minutes to complete this task.

To conclude the study, we asked the participants a few questions about their experience with the browsing and searching interface.

TASK 4.2 Active vs passive voice when writing the methodology

Read the following pairs of extracts. For each pair of extracts say which you prefer and why.

Extract A1

Participants were given an off-the-shelf installation of Flex Builder, pre-loaded with three project files. Participants in the control condition were provided with Firefox Web browser; they were asked to use the Adobe Community Help Search engine to look for example code. Participants in the treatment condition were provided with Blueprint to search for code samples; they were not allowed to use a web browser.

Extract A2

We gave participants an off-the-shelf installation of Flex Builder, pre-loaded with three project files. We provided participants in the control condition with Firefox Web browser; we asked them to use the Adobe Community Help Search engine to look for example code. We provided participants in the treatment condition with Blueprint to search for code samples; we did not allow them to use a web browser.

Extract B1

Parsing
The parser I used in this project is the part for abstracting phrases out of a sentence and storing them in a database. I used the phrases for searching related tweets from Twitter instead of searching the entire headline of news to increase the volumes of tweets returned. However, I found it difficult to get the high related phrases out of the headlines due to the nature of language. Therefore, I introduced three simple techniques to improve the quality of the keyword abstraction.

Extract B2

Parsing
The parser used in this project is the part for abstracting phrases out of a sentence and storing them in a database. The phrases were used for searching related tweets from Twitter instead of searching the entire headline of news to increase the volumes of tweets returned. However, it was difficult to get the highly related phrases out of the headlines due to the nature of language. Therefore, three simple techniques were introduced to improve the quality of the keyword abstraction.

TASK 4.3 Identify the use of tense for various methods

Read an adapted method section in a final report written by a previous student below. The tenses have been highlighted in different colours as shown below.

Read an adapted method section in a final report written by a previous student below. The tenses have been highlighted in different colours as shown below.

Present tense Past tense

1. Which tense is used to describe specific methods that have already been completed?

2. Which tense is used to explain working principles / describe standard methods used in the field or industry?

3. Imagine this is your FYP and you are writing about the proposed methodology that you will carry out later this semester. What changes would you make to the tenses?

Text 2 [2]

TITLE: Ranking News Headlines by Crowdsourcing
METHODOLOGY (on the News RSS feed and Parsing)

RSS feed

RSS (RDF Site Summary or Really Simple Syndication) **is** a family of web feed formats which **is** generally used to publish frequently updated works such as blog entries, news headlines, audio, and video in a standardized format. The feeds **allow** people to get the latest headlines as soon as they are published, without having to visit the original news websites. In the project, news headlines **were** retrieved from the news feed provided by news websites. Since we **were** focusing only on business and technology stories, only related headlines **were** collected.
[text removed]

Parsing

The parser used in this project **is** the part for abstracting phrases out of a sentence and storing them in a database. The phrases **were** used for searching related tweets from Twitter instead of searching the entire headline of news to increase the volumes of tweets returned. However, it **was** difficult to get the highly related phrases out of the headlines due to the nature of language. Therefore, three simple techniques **were** introduced to improve the quality of the keyword abstraction.

4.2 Reporting technical information using graphics and figures

Graphics and figures can frequently illustrate sophisticated concepts and phenomena more clearly than words. They are indispensable in technical writing.

There are three main components in referring to technical illustrations (tables and charts inclusive).

Three components in referring to technical illustrations

1. introducing the illustration (figure or table number – this tells the reader when to refer to the illustration)
2. describing the illustration (what is it about?)
3. offering comments on the illustration (optional for a simple procedure / visual)

Including these components ensures coherence between the illustration and the related text.

TASK 4.4 Identify the three components in presenting a technical illustration

Read the figure and the related text in a technical report titled "Image-based Exploration of Massive Online Environments". Identify the three components mentioned in this section.

Hint: pay attention to the evaluative language (positive or negative) signals and comments from the writer.

Text 4 [3]

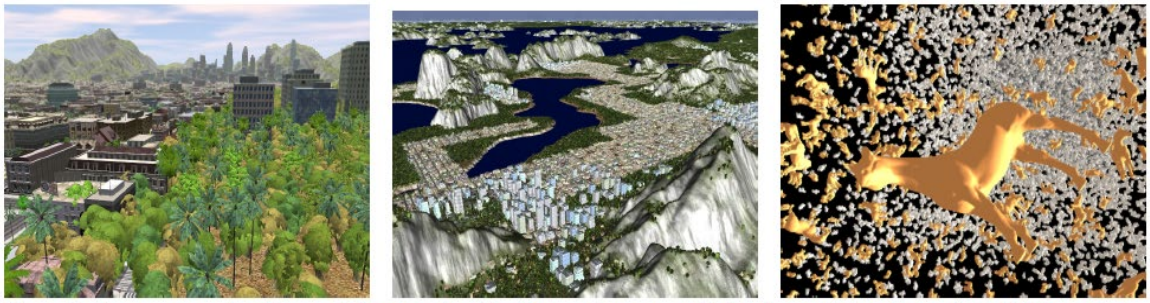


Figure 1: *Two scenes rendered interactively over a broadband network on commodity hardware. (Left and center) A 25-billion polygon, 10,000 km^2 Earth-like scene. (Right) A fully three-dimensional scene with 3 billion polygons.*

The computer graphics research community has advanced a variety of promising approaches to the display of large environments. Our application scenarios feature a combination of extreme scale and complexity, large numbers of distinct heterogeneous objects, and high visibility (see Figure 1). These factors conspire against exclusive reliance on visibility preprocessing, geometric simplification, and impostors. They also limit the applicability of approaches specialized for urban models and terrains.

TASK 4.5 Identify the three components for another technical illustration

Read the technical report titled "Example-Centric Programming: Integrating web search into the development environment." Answer the three questions below:

1. Identify the three components mentioned in this section.
2. Comment on whether you can follow the explained scenario.
3. What are the similarities and differences in this illustration and the one in the previous task?

Text 5 [1]

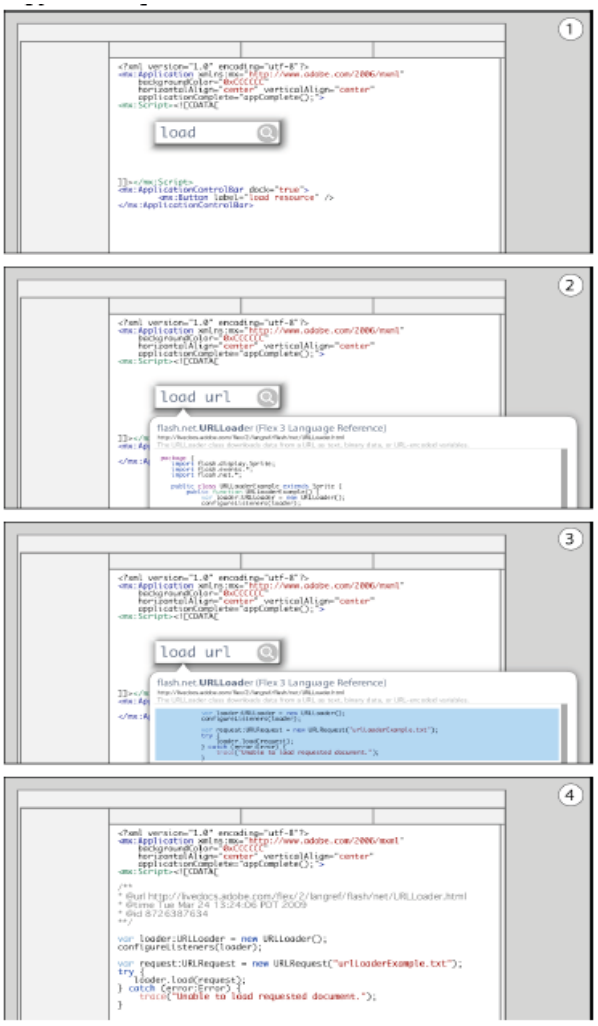


Figure 2. Example-centric programming with Blueprint. To initiate blueprint, the user press a hotkey to initiate a search; a search box appears at the cursor location (1). Searches are performed interactively as the user types; example code and running examples (when present) are shown immediately (2). The user browses examples with the keyboard or mouse, and presses *Enter* to paste an example into his project (3). Blueprint automatically adds a comment containing metadata that links the example to its source (4).

SCENARIO: DEVELOPING WITH *BLUEPRINT*

Jenny is a web programmer at a power utility that is about to launch a campaign encouraging individuals to lower their power consumption. She is prototyping a web application for customers to compare their daily power consumption to average levels. Her functional prototype should: *load user data* from a server into the client application; presses a hotkey to invoke the *Blueprint* search dialog; a search box appears next to her cursor (see Figure 2, step 1).

This interface extends auto-completion techniques to presents entire blocks of example code as results instead of variable and method names. She types *URLLoader*, the name of the main class associated with this task. *Blueprint* knows the language and framework version she is using and returns appropriate examples (step 2). These results are presented below the search box. She flips through the first few examples (step 3) and sees one that creates an XML object out of the data that is returned. She presses *Enter*, and the code is pasted in her project (step 4).

Along with the code, *Blueprint* adds a special machine- and human-readable comment that records the URL of the source and the date of copy. Every time Jenny opens this source file in the future, *Blueprint* will check this URL to see if the original example has changed (*e.g.,* if a bug is fixed), and will notify her when it does. She runs her code in Flex's debugger to confirm the XML has loaded and to inspect its format.

In addition to presenting the mentioned components and an illustrative figure caption, it is also useful to consider some of the general rules on including graphics in documentation listed below. [4]

Always make reference(s) to the figure and table in the text; for example, Figure 2 shows Example-centric programming with *Blueprint* or a search box appearing next to the cursor (See Figure 2, Step 1).

General rules on including graphics

1. Always describe and comment on the figure and table. Data do not speak for themselves; you have to speak up for them ☺.
2. Number the diagrams and tables separately starting from *1*.
3. In general, put the figures as close as possible to the reference text.
4. Include scale, legends, and units if applicable.
5. Use a readable font size.
6. Reserve sufficient white space above and below the figure or table.
7. Include an illustrative caption or title for your figure or table. In general, put figure captions below the figure and table captions below the table.
8. Avoid excessive highlights of figure and table captions, such as bolding, capitalization, and italics. These features can be intrusive and distracting.
9. If two figures are highly related, combine the two figures. If there are too many elements to focus on, then label the highly related figures as Fig. 1a and Fig. 1b instead of Fig. 1 and Fig. 2.
10. If the figure or table is obtained from a source different from your own data or findings, cite the source.
11. Figures copied from a secondary source must be correctly referenced. Give the source of the diagram if it is from a published source.

TASK 4.6 Identify good points and pitfalls in using technical illustrations

Work as a group. Look at some of the snapshots of illustrations in the reports written by previous students. Identify the good points and pitfalls. Name additional rules that you think are important.

Beijing Zhengzhou Changsha Guangzhou Hung Hom

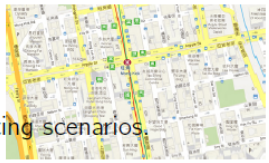


Figure 1. Packing can associate with various routing scenarios.

The problem comes from a combination of two NP-hard problems: (1) bin packing – packing two-dimensional items within a certain area called the loading space of a vehicle (or a bin), so that the least number of bins are used given that the items do not overlap; and (2) vehicle routing – assigning routes to each vehicle so that the distance traveled by that vehicle is minimized.

Snapshot 1

4.2.2 Member Registration

There is a registration icon at the right hand corner. For the registration form, users need to type in their email address, display name, and input the password twice for confirmation. A validation form tool is used. It validate if the users have input all the required information, it also validate the email's format, the length of the password must at least 6 characters and if the retype password matched. By using the AJAX, the validation tool can also check if the email and display name existed in the database already. In the registration form, users also need to verify if they are the salon's owner, as this determines their membership type with distinct features.

Registration form with validation tool

Snapshot 2

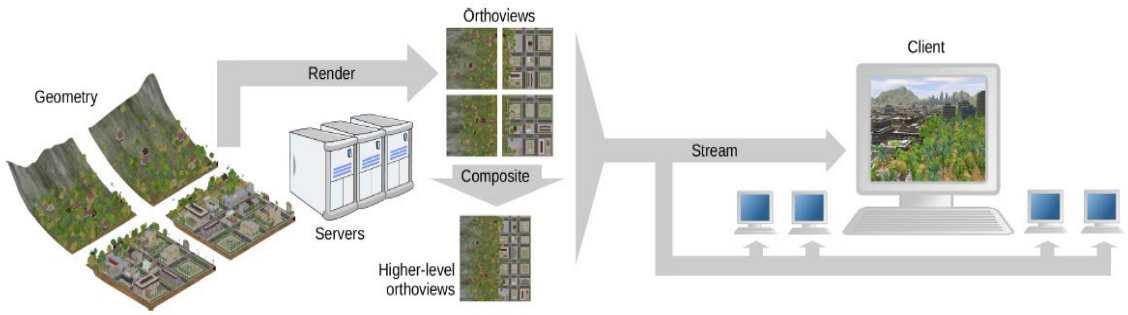


Figure 2: *An overview of the system.*

Our system has a distributed client-server architecture, in which each client renders its view of the world using geometry for nearby objects and a set of depth images for more distant ones.

Snapshot 3

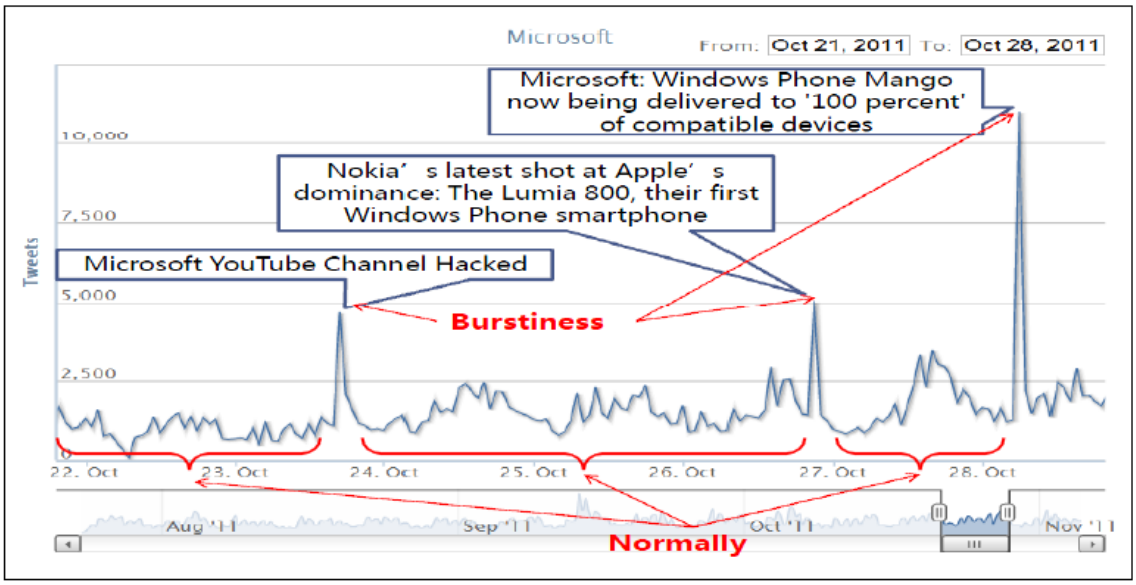


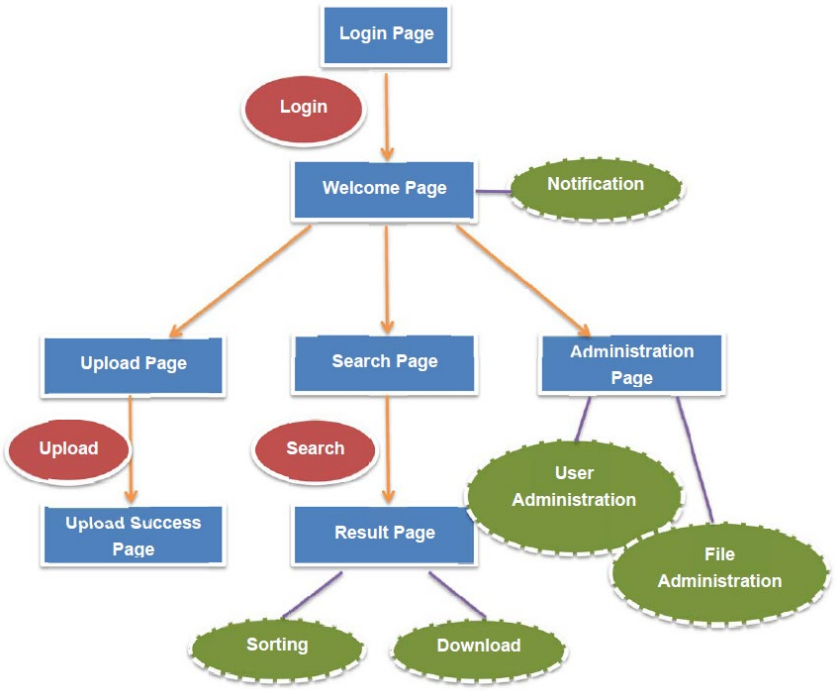
Figure 1: A graph showing how the word "*Microsoft*" appeared in Twitter. We used the Search API to collect all the tweets mentioning about *Microsoft* over the past 1 hour. Using this procedure, we kept getting the number of tweets in every hour for a week from October 21 - 28, 2011 and plotted against time.

Snapshot 4 (accompanied text follows)

For example, in Figure 1, the third "peak" comes after Microsoft turning its Windows Phone Mango being to "*100 percent*" of compatible devices. If we want to know whether people is talking about "*Microsoft*" a lot and very sudden, we have to know how many people talking about the topic as well as how many people talk about it before the news published. We call it the threshold of the topic, which is mean of the usual number of tweets in when nothing happens. By comparing these 2 information, we can know how burstiness of the topic is and learn the public attention (The term "burstiness" is used to describe this kind of sudden in this project).

3.1. Main Flow of System

The following graph shows the main flow of our system.



Snapshot 5

TASK 4.7 Apply your knowledge in using technical illustrations

Identify the main theme and pitfalls of one of the illustrations in your project. Write a short paragraph to describe and comment on the illustration. Do this for your Progress Report 1.

4.3 The assertion-evidence approach in oral presentations

The assertion-evidence approach has been designed specifically for technical and scientific presentations and is popular in the fields of science, medicine, and engineering. Your class teacher will demonstrate this approach through an interactive presentation.

TASK 4.8 Assertion-evidence approach: Spontaneous speech

The two extracts below are from the oral presentation and the written report of a student FYP on the theme of Virtual Reality.

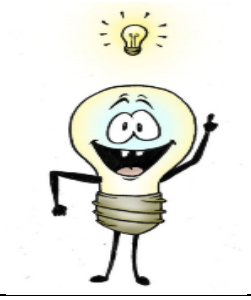
- What are the major differences between written and spoken language? Try to list at least 3 differences by comparing the extracts.

Spoken Text	Written text
So now I am going to move on to discuss the features of the product. I used Unity and Steam VR as interfaces for building this application. The first feature of this product that users will see is the main menu which acts a lobby. As you can see, the main menu has 3 black spherical objects which are the teleport points. Users can use them to travel around and switch between different scenes. From this main menu, you can access 3 different laboratories, each with their own experiment. The first is the Millikan's Oil Experiment. In this experiment, you use your hand controllers to guide different types of oil droplets into the chamber and experience the different outcomes in first person. The second laboratory has the Moon Experiment in which you can try to grab, drop and throw whatever objects there are in the environment. There are 2 planets provided for this experience: Moon and Mars. This way you can experience the difference between their gravitational forces. Last but not the least is the car stimulator. I am sure most of you would have some experience with a car stimulator game. It is a driving game which will be over if your car slides off the road into the mud area. The purpose of this is to present the idea of skidding and centripetal forces to users.	**4.1 Main Menu** The purpose of the main menu is to introduce the application and allow players to choose between different laboratories to enter. It is built in a room and acts like a lobby which consists of five stations. Illustrated in Figure 4.1. The first feature is teleportation, it is implemented using the Steam VR package. A few components are required to be added to the scene for the function. The player prefab from the Steam VR interaction system is added to the scene. Then the classes *Teleport*(), *TeleportPoint*(), and *TeleportArea()* are added also. The aim and basic introduction of the VR Physics laboratory are provided to the users with a "Learn More about the Lab" teleport station in the lobby. The station has a table with a few plane objects acting as papers which will be highlighted yellow when selected by the player to notify the selection and the player can then grab it to view the content. The players can choose to enter different laboratories by teleporting themselves to the portal gates. This is implemented by *loadscene()* function to the *TeleportToScene()* function to the teleport point prefabs. **4.2 Millikan's Oil Experiment** This laboratory demonstrates the experiment performed by Robert A. Millikan and Harvey Fletcher in 1909 to estimate the charges in an electron. The experiment includes some principles of ionization, electric field, and gravitational force. In the experiment, charged oil droplets pass through electric fields in the chamber. The result of the experiment is that the oil droplets may react differently when applied to the same electric field according to their masses. Differences in masses will affect the magnitude of their downward forces. Below is Figure 3.2 provided by the School of Physics at the University of College Cork illustrating the experiment. It is hoped that students can learn about topics like electric fields and gravitation in a more unconventional way.

4.4 Over to you

Technical communication is perhaps distinct from other forms of communication in its emphasis on explaining methodology and using charts and graphs. In a dynamic field of computer science, it is important to work out the nature of your project (experiment vs development) and the underlying concept (such as those identified in analogies) before 'speaking out' for technical data presented in charts and graphs.

TASK 4.8 Reflect on this unit



Key points to remember

- Methodologies are presented differently in experimental versus developmental projects.
- Use different tenses to illustrate methods standard in the field (Simple Present), proposed methods (Simple Future), or those already specifically administered (Simple Past).
- Generally use passive voice to focus on the procedures.
- When using a technical illustration – introduce it, describe it, and then comment on it.
- Always revisit the rules of and common pitfalls in using a technical illustration.
- Focus on the underlying concept and logic flow by using pseudo-codes and analogies.

Homework and Preparation for the next session

- Apply what you have learnt about methodologies and the use of graphics in writing your progress report 1.

References

[1] Brandt J, Dontcheva M, Weskamp M, Klemmer SR. Example-Centric Programming: Integrating Web Search into the Development Environment. *Stanford Computer Science Technical Report.* Stanford University; 2009.

[2] Adapted from student texts.

[3] Chaudhuri S, Horn D, Hanrahan P, Koltun V. Image-Based Exploration of Massive Online Environments. *Stanford Computer Science Technical Report*. Stanford University; 2009.

[4] Sorby A, Bulleit W. *An Engineer's Guide to Technical Communication*. Ohio: Pearson Prentice Hall; 2006.