Background Research Section

# Raspberry Pi and CECIL

In the year 2012, the Raspberry Pi Foundation developed a single-board computer with the intention of promoting the teaching and learning of Computer Science in schools. This device, known as Raspberry Pi (RPi), constitutes a low-cost computing and interfacing machine to external devices, which allows for greater accessibility in designing integrated systems. However, RPi does not yet comprise the necessary scaffolding tools in interfacing and controlling the system. Consequently, customers find it difficult to quickly, efficiently and enjoyably learn the concepts necessary to understand and work with RPi. Currently, there are several projects being undertaken for the development of such scaffolding tools.

# Literature review

The Raspberry Pi foundation and its affiliate companies strive for high levels of customer satisfaction. Consequently, new and current scaffolding tools must be developed in a highly usable and accessible fashion.

## Striving for usability

### “The design of everyday things”

Norman, in his seminal book, *The Design of Everyday Things,* approaches product design on a generic and practical way. He identifies ways in which good product design can be used to anticipate problems and overcome them before they arise.  Although his work was more generic than just software design, his approach is particularly important to creating high levels of usability. He points out ways in which shortcomings in design can generate frustration and confusion for the user. (1)

Within the gulfs of execution and evaluation the user makes many mistakes – “If an error is possible, someone will make it”. (1 p. 36) The gulf of execution is defined as the difference between the intentions and the allowable actions of a system. It is measured by the user’s ability to achieve a set goal without extra effort. Whereas the gulf of evaluation reflects the amount of effort a person must exert to interpret the state of a system, and consequently determine how successful they have been in reaching the set goal. (1 p. 51) In addition, much human behaviour is done subconsciously, through pattern matching, and proceeds rapidly and automatically without effort. Therefore, users must be supplied with a simple system that doesn’t induce any struggle. Unfortunately this is very rarely the case. Several user needs must be taken into consideration when designing a tool, which many computer systems designers seem to be oblivious to. Ergo, problems are frequently amplified, leading to the “tyranny of the blank screen”, meaning the user is left with a confusing system in which no sensible action seems feasible.

However, there are pieces of software, such as Facebook which have shown to be highly usable systems – “Programs and systems do exist that have shown us the potential; they take the user into account, and they make it easier for us to do our tasks – pleasurable, even. This is how it ought to be”. (1 p. 178) In order to achieve such potential in this project, the software implemented must accord with five design principles: visibility, constraints, affordances, natural mappings and feedback. (1 p. 181)

Firstly, the designer must take into account both the knowledge and experience users have of similar systems and their own knowledge as an expert in the field, never underestimating either of them. Secondly, it is important to simplify the structures of tasks as much as possible. The aforementioned five principles describe the ideal design for task simplicity in a system.

Norman states that “Visibility indicates the mapping between intended actions and actual operations”. (1 p. 8) Ergo, this principle demonstrates the connection between the users’ mental models of a system and the system itself. These are models people have of products with which they interact and that allow the user to predict the effects of their actions. They are formed through experience, training, instruction and the perceiving of a system’s available procedures and visible structure. Therefore, it is of great importance for a system’s design to not be incoherent or inappropriate, as this might result in the user operating it by rote, blindly. (1 p. 17&179) Useful conceptual models are created when, within a software piece, the controls and parts needed are visible and the implications are clear. The clarity of implications comes from clues on how things work in the system: constraints, affordances and mappings. (1 p. 12)

“Design is the successive application of constraints until only a unique product is left”. (1 p. 158) This quote refers to the power of constraints on letting the user determine readily the proper course of action. There exist several types of constraints:

* Physical constraints, which rely upon the physical properties of a product for their operation. For example, the action of starting a car can be constrained by the failure of some physical property of the car. A constraint of this kind is also known as a forcing function. (1 p. 84 & 132)
* Semantic constraints, which rely upon the user’s knowledge of the situation and the world to control the set of possible actions. (1 p. 85) For example, one always knows that a search button or box is for allowing the user to search for something.
* Cultural constraints, which rely upon accepted cultural conventions. For example, in one culture it would make sense to build a car with a steering wheel on the right side, whereas in another culture, where they drive in the opposite side of the road, it would make sense to put the steering wheel on the left side of the car. (1 p. 85)
* Logical constraints, which rely on the logical relationships between spatial and functional layout of components and the things they are affected by. (1 p. 86)

Affordances are quite similar to constraints in the sense that they constrain operations; however, they provide strong clues on how to operate things rather than prohibiting certain actions. Affordances refer to the perceived and actual properties of a system. For example, a chair is physically made to afford support; therefore, the user is informed that it is for sitting on. (1 p. 9)

Mappings stand for the relationship between a system’s controls and their actions. These are easily learned and remembered, however they can also cause many difficulties in understanding a system. (1 p. 23) For example, a user would expect a navigation menu to be placed at either the top or left hand side of a page. If a user were to find a navigation menu at the bottom of a page they are likely to be confused. In addition, a good mapping is derived from a successful combination of visibility and feedback.

Feedback is the process of sending back to the user information about the consequences of actions performed in the system (i.e. what result has been accomplished). Hence, good feedback should give an immediate and obvious effect for an action performed. (1 p. 27 & 99)

Several systems attempt to follow the five principles, described previously, in order to achieve an ideal design; however, this can be a time-consuming, expensive and complex procedure. Ergo, all systems, including CECIL, present faults. Nevertheless, these faults can be fixed and consequently the system improved. In this project, enhancement methodologies were investigated to optimise CECIL’s interface.

### “Don’t make me think”

Steve Krug utilises a different approach to software usability from the previously described. Its work, based on the perspective of a website consultant, constitutes a series of web design enhancement methodologies. As described by Krug, “on the internet, the competition is always just a click away, so if you frustrate users they’ll head somewhere else”. (2)

GET ALL NOTES HERE!

### Comparison of current with other educational technologic resources

COMPARE WITH AN EDUCATIONAL RESOURCE FOR COMPUTING BASED ON TECHNOLOGICAL IMPLEMENTATION

## Accessibility

### IBM – definition of software accessibility

“An accessible software product is software that can be used effectively by users with certain kinds of disabilities. Accessibility involves converting software that can only be used by people without disabilities into software that can be used by people with and without disabilities alike.” (3)

A BETTER INTRODUCTION TO ACCESSBILITY

### The Java Accessibility API

(4)

### “Engineering Software for Accessibility”, Microsoft Corporation

(5)

### Comparison of current with other educational technologic resources

COMPARE WITH AN EDUCATIONAL RESOURCE FOR COMPUTING BASED ON TECHNOLOGICAL IMPLEMENTATION

# References

1. **Norman, Donald A.** The Design of Everyday Things. New York : Basic Books, 2002.

2. **Krug, Steve.** *Don't Make Me Think A Common Sense Approach to Web Usability .* Berkeley, California : s.n., 2006.

3. **IBM.** Accessibility: the definition and the disabilities it covers. *IBM Corporation Website.* [Online] 2011. [Cited: 15 October 2013.] http://pic.dhe.ibm.com/infocenter/jviewmap/v8r8/index.jsp?topic=%2Fcom.ibm.ilog.jviews.defense.doc%2FContent%2FVisualization%2FDocumentation%2FJViews%2FJViews\_Defense%2F\_pubskel%2Fps\_usradvfwork1429.html.

4. **ORACLE.** Java Accessibility API (JAAPI) Programmer's Guide. *ORACLE Java SE Documentation.* [Online] 2013. [Cited: 15 October 2013.] http://docs.oracle.com/javase/7/docs/technotes/guides/access/jaapi.html.

5. **Microsoft Corporation.** *Engineering Software for Accessibility.* s.l. : Microsoft Press, 2009.