2015

Teaching Universal Design in Computer Science

Damian Gordon
Technological University Dublin, Damian.Gordon@tudublin.ie

Ciaran O'Leary
Technological University Dublin, ciaran.oleary@tudublin.ie

TEACHING UNIVERSAL DESIGN IN COMPUTER SCIENCE

Damian Gordon,
School of Computing,
Dublin Institute of Technology,
Kevin Street, Dublin 8, Ireland.

email: Damian.Gordon@dit.ie

Ciarán O'Leary,
College of Sciences and Health,
Dublin Institute of Technology,
Kevin Street, Dublin 8, Ireland.

email: Ciaran.OLeary@dit.ie

Abstract

The principles of Universal Design developed at North Carolina State University in 1997 are well-known and frequently cited. When teaching Universal Design in Computer Science the principles are frequently used, and the focus is generally on the user interface elements of design, as the principles can easily be appreciated in the context of ideas such as User Experience (UX), Human-Computer Interaction (HCI), or Visual Design (which uses *Wizard of Oz prototyping*). User interface design considers how a user will experience the software, and recommends that programmers ensure that the interface is as simple and efficient as possible, in terms of accomplishing the users' goals. An often overlooked element of Universal Design in software design is to consider the software itself, on how it is built, and how it is formatted, using the lens of Universal Design. Given that the reality is that most code will be modified by a developer who may be unknown to the original developer, it is important that code is designed (both in terms of build and format) in such a way that it is future-proofed and therefore universally designed.

Introduction

This research was initiated as a result of a new programme of study started in Dublin Institute of Technology's School of Computing in September 2015. The new programme is a BSc in Information Systems and Information Technology (Course Code: DT255), and uses a blended learning (Oliver and Trigwell, 2005) delivery model. In this case, some modules will be taught in a traditional bricks-and-mortar classroom, whereas other lectures will be delivered fully on-line.

Delivering lectures fully on-line presents significant challenges for both lecturer and student. Diana Laurillard's Conversational Framework (1993) suggests that lecturer-student interaction is of paramount importance in teaching, and that is important to recognise that both the lecturer and student has a wide-ranging existing set of concepts in their heads, and the greater the difference between their sets of concepts, the more difficult the teaching process becomes, and the more consideration that the lecturer has to give both to the construction of the learning environment, and the nature of the activities that they get the students to do (see Figure 1).

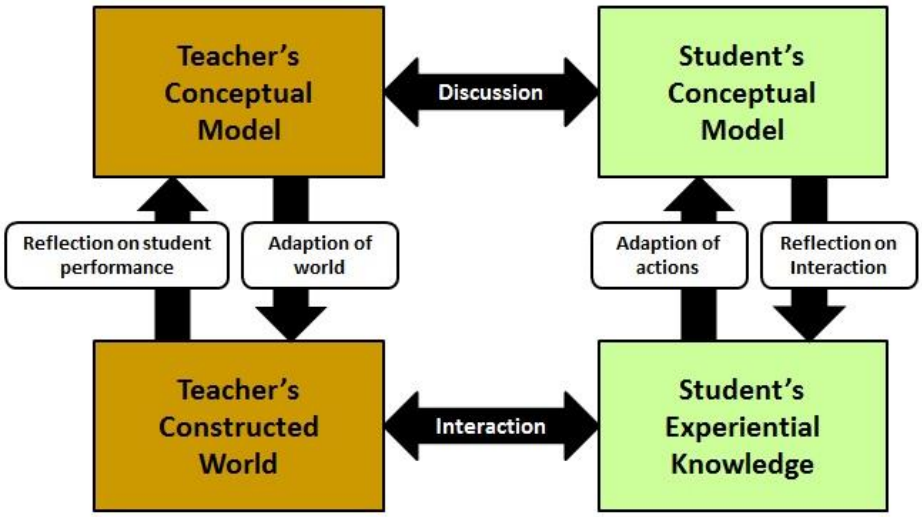

Figure 1. Laurillard's Conversational Framework

An on-line module also requires a significant degree on intrinsic motivation and maturity on behalf of the participating students. Gilly Salmon (2002) proposed a Five-Stage Model of eLearning to describe the levels of maturity a student goes through in an eLearning environment which will be very important in this program:

- **Stage 1: Access and Motivation** – At this stage the student is new to a learning environment, and there will be a few technical issues at first, therefore the lecturer must be welcoming and encouraging.
- **Stage 2: Online Socialization** – At this stage the student is starting to learn more about their learning environment and is linking with fellow students, therefore the lecturer must act as a moderator and facilitator.
- **Stage 3: Information Exchange** – At this stage the student is confidently sending and receiving messages from other students, and acting as their own moderator, therefore the lecturer focuses on delivering learning materials and e-tivities.
- **Stage 4: Knowledge Construction** – At this stage the student is generating their own knowledge and contributing it to the group, therefore the lecturer acts as the overall architecture of the contributions into a cohesive whole.
- **Stage 5: Development** – At this stage the student have taken ownership of their work, and are able to apply them in their own context, therefore the lecturer steps back, but is available for questions and answers.

Thus, in this programme the students will be guided through the on-line aspects of the programme in a manner that will allow them to develop and mature their online learning skills.

Pam Moule (2007) extends and challenges Salmon's model in developing the eLearning Ladder, which acknowledges a wider range of learning activities, and considers the level of ICT skills of all the parties involved have, as well as considering issues such as technical support and access (see Figure 2).

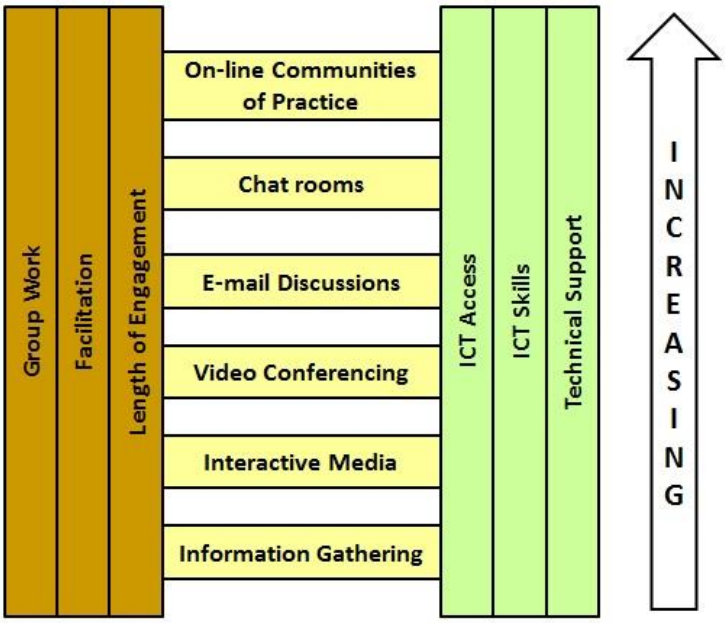

Figure 2. Moule's Ladder of eLearning

To bring together all of the above, Robert Gagné's Nine Events of instructional Design will be the framework used for each of the on-line lessons. Gagné suggests that the first stage of the instructional design process is to formulate a clear learning goal, following that he provides a step-by-step model of how to undertake a lesson, including; how to present the knowledge, how to demonstrate the skills, and how to assess the learning.

The steps Gagné suggests start with requiring the lecturer gaining the attention of the students (stage 1); this will be done differently in a traditional classroom than it will be in an on-line environment. Next the lecturer must describe the learning goals of the lesson (stage 2); this will be similar in the traditional and online settings. Following this the lecturer must highlight previous lessons that tie in with the current lesson (stage 3); this will be similar in the traditional and online settings. The next stage will be to present the learning materials to the students (stage 4); this will be done differently in a traditional classroom than it will be in an on-line environment. The lecturer will then provide guidance to the students as to how to understand the material (stage 5); this will be done differently in a traditional classroom than it will be in an on-line environment. Next the lecturer will ask the students to perform a relevant task (stage 6); this will be done differently in a traditional classroom than it will be in an on-line environment. From here the lecturer will provide feedback (cf. Hattie) (stage 7); this will be done differently in a traditional classroom than it will be in an on-line environment. For this the lecturer will assess the overall learning the students have achieved (stage 8); this will be similar in the traditional and online settings. Finally the lecturer will check what the students have retained after a long period of time (stage 9); this will be similar in the traditional and online settings.

On-line Module: Programming and Algorithms

One of the modules whose lectures will be delivered fully online is called "Programming and Algorithms" and focuses on an introduction to the design and development of software. The module is designed so that at the end of the module, the students will be able to:

- Design and write computer elementary programs in a structured procedural language.
- Use a text editor with command line tools and simple Integrated Development Environment (IDE) to compile, link and execute program code.
- Divide a computer program into modules.
- Test computer programs to ensure compliance with requirements.
- Implement elementary algorithms and data structures in a procedural language.

The students visit the module webpage[1] (see Figure 3) and view each week's videos, and read that week's PowerPoints and Code Samples (see Figure 4). Following this they are required to do activities on the discussion board, and do a Laboratory once a week that will include exercises on topics that they have reviewed in that week.

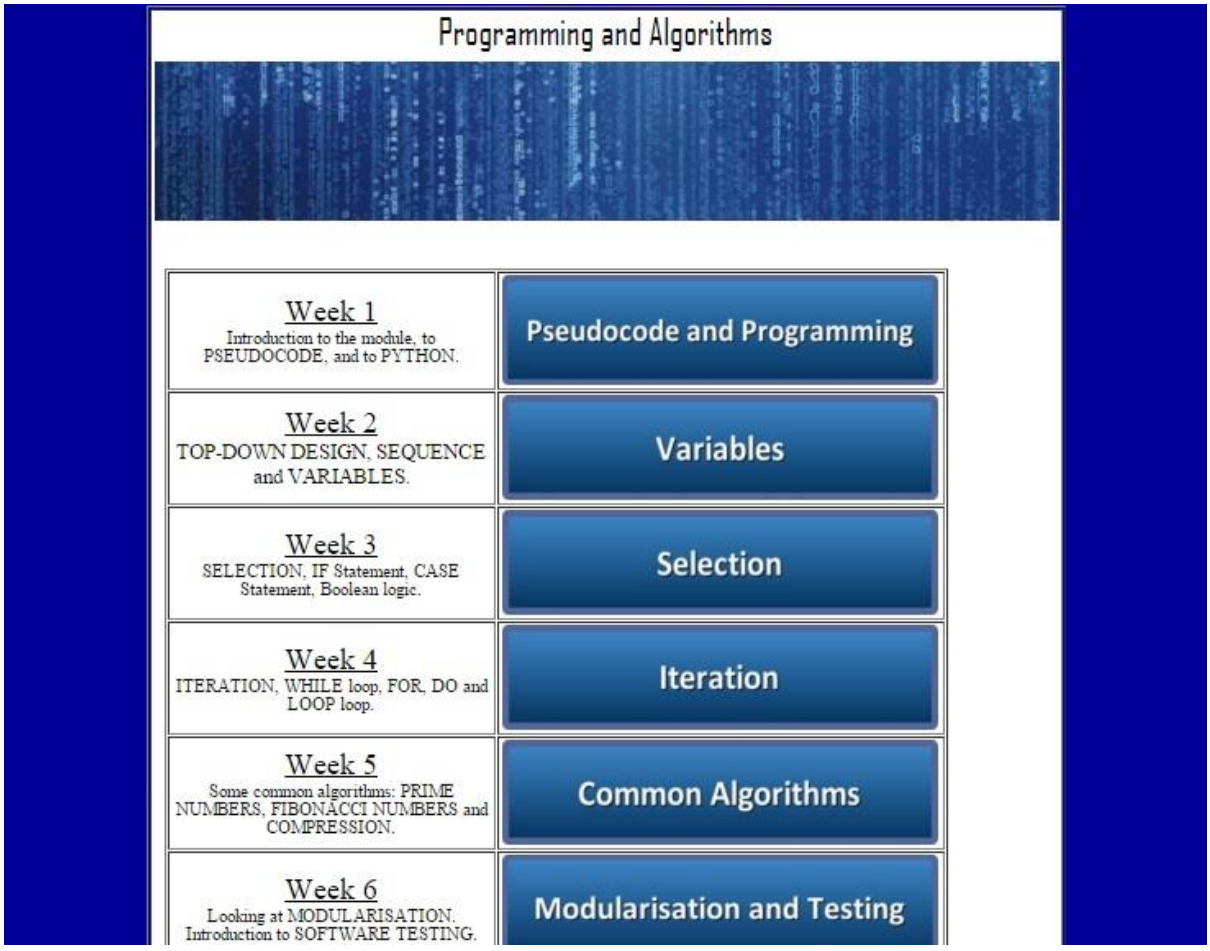

Figure 3. Module Website

[1] http://www.damiantgordon.com/python

Figure 4. Lesson Webpage

Since the design of software is an integral part of the module, one of the key topics being taught and being discussed in this module is Universal Design, focusing particularly on the principles of Universal Design, as presented by researchers at North Carolina State University in 1997, which represent a clear and coherent set of ideals for the design of products, services and environments (The Center for Universal Design 1997). Rather than being an end-point, these principles should be recognised as a starting point, providing the first generally agreed set of principles defining Universal Design (See Appendix A). Over time the principles have been questioned and challenged, as should be the case for any set of principles, with competing versions emerging occasionally from the literature, most notably Erlandson's principles (Erlandson 2007).

The sixth and seventh principles are clearly less relevant to software development than the others, but in a bricks-and-mortar classroom setting, with discussion,and reflection, it is possible to situate these principles in a Computer Science setting, whereas in the on-line delivery, it is necessarily to be more directed, therefore a new perspective on the principles had to be developed, including seeing the principles both from a users' and a developers' point-of-view.

Layering the Principles

The authors have argued previously that the principles can be viewed as consisting of three semiotically distinct layers, with principle 1 (Equitable Use) residing in a layer by itself as the overall philosophy of Universal Design. Following this, principles 2-5 (Flexibility in Use, Simple and Intuitive, Perceptible Information, and Tolerance for Error) in a separate layer which describes some of the principles that must be considered to achieve the overall philosophy. Finally principles 6-7 (Low Physical Effort, Size and Space for Approach and Use) are domain-specific principles in the Built Environment which describe how to achieve the previous layer's goals (O'Leary and Gordon, 2009; Gordon and O'Leary, 2011).

Layer	Principle	Description
Layer 1	Equitable Use	Overriding Philosophy
Layer 2	Flexibility in Use	General Principles for Realising Philosophy
	Simple and Intuitive	
	Perceptible Information	
	Tolerance for Error	
Layer 3	Low Physical Effort	Principles for Realising Philosophy within the Built Environment Domain
	Size and Space for Approach and Use	

Table 1. The Principles of Universal Design

Level 1 is comprised of a single principle which describes the overriding philosophy of Universal Design. Any design should be evaluated for its adherence to this principle, across all domains. It is a high level summary and clear explanation of the philosophy of Universal Design.

Level 2 is comprised of the next four principles. These remain, in our consideration and again without considering the guidelines which accompany them, domain general, and serve as specific means to arrive at the promise of the first principle. Anything which is designed in any domain for any problem should *be flexible in use*, *be simple and intuitive*, present *perceptible information* and incorporate *tolerance for error*. These domain general principles may require a moderate rewording, but as they stand at present, they represent useful principles against which any designs can be evaluated.

Level 3 from the above list are domain specific. *Low physical effort* and *size and space for approach and use* are applicable only to specific problems, most notably in the design of products and environments. These represent principles for realizing the *Level 1* principle in a specific domain.

Visually we can place Principle 1 as the overall goal, Principles 2-5 as the pillars, and Principles 6-7 as the foundations within a specific discipline:

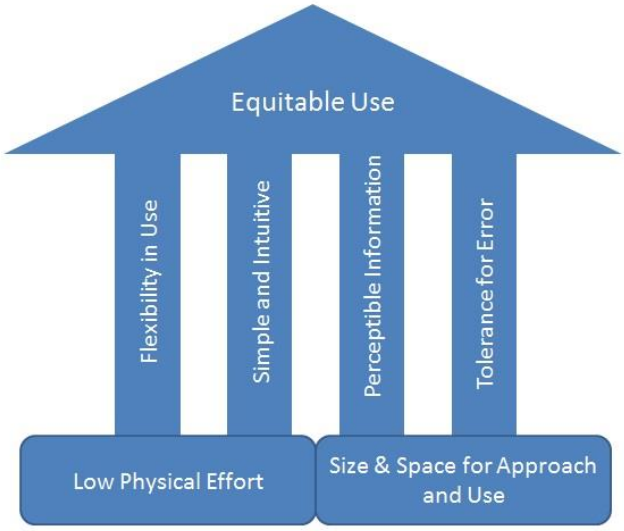

Figure 5. The Seven Principles of Universal Design

Situating the Principles in the Computer Science Discipline

To ensure that the principles are relevant and useful to Computer Science students, it is clear that Principles 6 and 7 need to be changed (*localised*) to issues in the Computer Science domain. To achieve this we will examine the notions present in defensive programming (or secure programming) which focuses on improving software and source code under three principles: (1) General quality, (2) Making the source code comprehensible, and (3) Making the software behave in a predictable manner. The first Principle is a general one, whereas Principle 2 will be recast to the more general goal of "Consideration for Users", and Principle 3 will be similarly recast as "Use of Patterns", therefore the Computer Science seven principles are as follows.

Layer	Principle	Description
Layer 1	Equitable Use	Overriding Philosophy
Layer 2	Flexibility in Use	General Principles for Realising Philosophy
	Simple and Intuitive	
	Perceptible Information	
	Tolerance for Error	
Layer 3	Use of Patterns	Principles for Realising Philosophy within the Computer Science Domain
	Consideration for Users	

Table 2. The Principles of Universal Design (in Computer Science)

As before visually we can place Principle 1 as the overall goal, Principles 2-5 as the pillars, and Principles 6-7 as the foundations within a specific discipline:

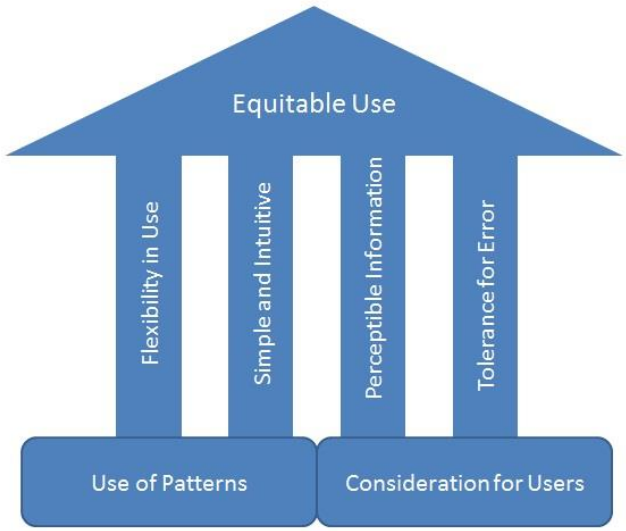

Figure 6. The Seven Principles of Universal Design (in Computer Science)

To add a bit more detail to these principles, it is worthwhile to suggest some guidelines that would be used to represent these principles in a Computer Science context, as follows:

Layer	Principle	Guidelines
Layer 1	Equitable Use	One product designed well for everyone.
Layer 2	Flexibility in Use	Configurable interface, adapts to user needs, variety of ways of achieving the same thing (e.g. hotkeys)
	Simple and Intuitive	Navigation pathways, metaphor, number of clicks, breadcrumbs, etc.
	Perceptible Information	The use of colours, use of clear language, etc.
	Tolerance for Error	Catching, preventing error, clear error messages.
Layer 3	Use of Patterns	Repeated themes in terms of navigation and functionality
	Consideration for Users	Understand the users' needs, consider personas, speak their language

Table 3. The Principles with Guidelines

In particular of note is the consideration of the use of personas, which is an increasingly popular approach to the design of interactive products and interfaces (Cooper, Reimann, and Dubberly 2003; Cooper 2004) as well as the software development process (Zimmermann and Vanderheiden 2005), primarily according to the modern agile development movement (Fowler and Highsmith 2001). Personas have also been explored by the author as a specific form of teaching approach (Gordon, *et al*., 2013).

However, this formulation fails to account for the difference between two distinct kinds of users – the End-Users and the Developers. The End-Users use the software in a very distinct way to the Developers who will have to modify, correct, and extend the existing software (in the same way that an extension on a house may not be undertaken by the original builder, or the features of a product might be extended by a new designer).

The End-Users only get to see the software executing, and therefore treat it as a black box (i.e. they don't see the computer programs), whereas the developers (who are a special instance of user) have to treat the code like a white box (i.e. they have to look into the code to modify it). Thus, we require two distinct set of guidelines for these distinct groups.

Layer	Principle	End-User Guidelines	Developer Guidelines
Layer 1	Equitable Use	One product designed well for everyone.	Algorithm set out to be reused in different languages, platforms etc.
Layer 2	Flexibility in Use	Configurable interface, adapts to user needs, variety of ways of achieving the same thing (e.g. hotkeys)	Modular, component based code. Well designed to be configurable etc.
	Simple and Intuitive	Navigation pathways, metaphor, number of clicks, breadcrumbs, etc.	Not using language-specific tricks, Library use.
	Perceptible Information	The use of colours, use of clear language, etc.	Documentation, Variable naming.
	Tolerance for Error	Catching, preventing error, clear error messages.	Secure, defensive programming practice.
Layer 3	Use of Patterns	Repeated themes in terms of navigation and functionality	Design patterns, and using the same coding approaches.
	Consideration for Users	Understand the users' needs, consider personas, speak their language	For the developer-user ensure modularity and extensibility,

Table 4. The Principles from the Users' and the Developers' point of view.

The New Principles in Detail

Based on these principles, it becomes possible to develop new guidelines for the principles, both for the End-user and the Developer. Interestingly a significant majority of the existing guidelines work perfectly well as End-User Guidelines (and are shaded grey), whereas the developer guidelines are all newly created, but strongly reflect the existing guidelines.

Principle 1: *Equitable Use*

End-User Guidelines	*Developer Guidelines*
A. Provide the same means of use for all users: identical whenever possible; equivalent when not. *B. Avoid segregating or stigmatizing any users.* *C. Make provisions for privacy, security, and safety equally available to all users.* *D. Make the design appealing to all users.*	A. Provide a range of IDEs and development environments. B. Ensure that all the necessary assistive technologies needed are provided. C. Provide versioning software, document backup facilities, and undelete features. D. Ensure the software is as readable and clear as possible.

Principle 2. *Flexibility in Use*

End-User Guidelines	*Developer Guidelines*
A. Provide choice in methods of use. *B. Accommodate right- or left-handed access and use.* *C. Facilitate the user's accuracy and precision.* *D. Provide adaptability to the user's pace.*	A. Provide a range of IDEs and development environments. B. Provide a range of input devices, e.g. keyboards, voice synthesis C. Provide code standards checking tools D. Develop in a modular, component based approach

Principle 3. *Simple and Intuitive Use*

End-User Guidelines	*Developer Guidelines*
A. Eliminate unnecessary complexity. *B. Be consistent with user expectations and intuition* [Navigation pathway, breadcrumbs] *C. Accommodate a wide range of literacy and language skills.* *D. Arrange information consistent with its importance.*[Metaphors] *E. Provide effective prompting and feedback during and after task completion.*	A. Implement features in common, expected ways, don't obfuscate. B. Be consistent with developer expectations. C. Accommodate a wide range of literacy and language skills. D. Arrange information consistent with its importance. E. Use software libraries when possible.

Principle 4. *Perceptible Information*

End-User Guidelines	*Developer Guidelines*
A. Use different modes (pictorial, verbal, tactile) for redundant presentation of essential information. *B. Maximize "legibility" of essential information.* *C. Differentiate elements in ways that can be described (i.e., make it easy to give instructions or directions).* *D. Provide compatibility with a variety of techniques or devices used by people with sensory limitations.*	A. Comment the code prolifically. B. Use clear variable names and module names. C. Build in help features into the code. D. Provide compatibility with a variety of techniques or devices used by people with sensory limitations.

Principle 5. *Tolerance for Error*

End-User Guidelines	*Developer Guidelines*
A. Arrange elements to minimize hazards and errors: most used elements, most accessible; hazardous elements eliminated, isolated, or shielded *B. Provide warnings of hazards and errors.* *C. Provide fail safe features.* D. *Discourage unconscious action in tasks that require vigilance.*	A. Develop software using the principles of defensive programming. B. Catch errors where possible. C. Give detailed and clear error messages. D. Avoid global variables, and modules that cause side-effects.

Principle 6. *Use of Patterns*

End-User Guidelines	*Developer Guidelines*
A. Provide repeated themes in terms of navigation. B. Provide repeated themes in terms of functionality. C. Provide standard screen formats. D. Provide visual cues.	A. Use software design patterns. B. Use the same coding approaches. C. Use the same naming standards for variables and modules. D. Use standard library functions.

Principle 7. *Consideration for Users*

End-User Guidelines	*Developer Guidelines*
A. Understand the users' needs. B. Consider the use of personas. C. Speak the End-users' language. D. Provide help features.	A. Develop modular code to help the developers B. Develop easily extensible code. C. Adhere to coding standards D. Comment complex elements of the code, and refer to design documents.

Discussion

If Universal Design can be said to be challenging, then it is challenging because it attempts to address a paradox at the heart of human existence. This paradox concerns the essential tension between the biology of the human being and their psychology. In terms of human biology the key to survival is diversity, the more diverse a species is, the more likely they are to be resilient to changes and challenges, this is why there is no one single type of person, we have diverse sizes, handedness, abilities, ages, eye colour, dexterity, etc. But in direct contrast to this is human psychology which strives for uniformity; the human mind is constantly exposed to a vast array of sensory information, and to cope with this, the mind simplifies the incoming information by using pattern-matching and categories to simplify that input. This type of simplification is also applied to people and results in viewing groups of people as an undifferentiated mass (in-group/out-group dynamics). So as a race biological diversity is imperative for survival, but as individuals we strive to find uniformity and simple categories, this Dualistic paradox is what makes Universal Design difficult, we know that there is massive diversity in the human population, and yet our brain tends to interact with the world in simplified, categorical ways, so for example when designing tools we tend to design for ourselves, (or our clan or our tribe) as opposed to designing for the evident diversity of the human population.

As stated in the abstract of this paper, we see the principles of Universal Design as fitting into three distinct layers. The first principle (Equitable Use) is an overriding philosophy statement; it is the goal of Universal Design. The next four principles (Flexibility in Use, Simple and Intuitive Use, Perceptible Information, Tolerance for Error) can be seen as the specific issues that should be addressed to achieve the first principle of Equitable Use. The final two principles (Low Physical Effort, Size, and Space for Approach and Use) are really not general principles, but rather refer specifically to the domain of architecture and built environment and would have less relevance in, for example, the field of the Universal Design of software.

From this simple analysis it is clear that the principles are an excellent starting point, but nonetheless only a starting point, for the design process it is important to consider the specific domain in which the principles will be utilised. In this research to scaffold the principles we extended and modified the principles to direct the development of a series of principles and guidelines for software development. The current version of the principles are merely another step along the way, the development and refinement of these principles and their guidelines is a continuous and living process, as more people contribute the them, they will grow to accommodate a wider range of perspectives.

Summary and Conclusions

In this research the development of a series of concrete guidelines based on a modified set of principles to address the needs of end-users and developers of software. The principles of Universal Design were used as the essential element to begin developing these new guidelines; from there the input of secure programming was incorporated into the concrete guidelines that were designed to be coherent and readable while at the same time addressing the needs of the diverse users of these guidelines. This approach proved to be highly successful and has resulted in a set of guidelines that are currently stable, but may result in additions or alterations to the existing guidelines after user testing.

References

Cooper, A. 2004. The inmates are running the asylum: Why high tech products drive us crazy and how to restore the sanity. Pearson Higher Education.

Cooper, A., R. Reimann, and H. Dubberly. 2003. About face 2.0: the essentials of interaction design. John Wiley & Sons, Inc. New York, NY, USA.

Gagne, R.M., Briggs, L.J., Wager, W.F., (1985) *Principles of Instructional Design*, Wadsworth

Gordon, D., O'Leary C., Universal Design is Sustainable Design, But How is it Measured? Seventh China - Europe International Symposium on Software Industry Oriented Education - CEIS-SIOE 2011, Northampton, UK, 23-24 May 2011

Gordon, D., Fennell, A., O'Leary, C., O'Connor, J., Persona-Based Teaching: A New Approach to Exploring the Dimensions of Universal Design using Personas and Scenarios, AHEAD 2013 Conference, Dublin, Ireland , March 12, 2013.

Erlandson, Robert F. 2007. Universal and Accessible Design for Products, Services, and Processes. 1st ed. CRC Press.

Fowler, M., and J. Highsmith. 2001. The agile manifesto. Software Development 9, no. 8: 28–35.

Laurillard, D.M. (1993) *Rethinking University Teaching: A Framework for the Effective Use of Educational Technology*, Routledge, London.

Moule, P, (2007) "Challenging the Five-Stage Model for e-Learning: A New Approach", *Research in Learning Technology*, 15(1).

O'Leary, C., Gordon, D., Universal Design, Education and Technology, In proceedings of the Ninth Annual Information Technology and Telecommunications Conference, Dublin Institute of Technology, Dublin, Ireland, 22-23 October, 2009

Oliver, M.,, Trigwell, K. (2005). "Can 'Blended Learning'be Redeemed?", *E-learning and Digital Media*, 2(1), 17-26.

Salmon, G. (2002) *E-tivities: The Key to Active Online Learning*, Kogan Page

The Center for Universal Design. 1997. The Principles of Universal Design, Version 2.0. Raleigh, NC: North Carolina State University.

Webb, N. L. 1997. Criteria for alignment of expectations and assessments in mathematics and science education. Council of Chief State School Officers and National Institute for Science Education Research Monograph.

Zimmermann, G., and G. Vanderheiden. 2005. Creating accessible applications with RUP. At http://www. ibm. com/developerworks/rational/library/jul05/zimmerman/index. html. August '09.

Appendix A: The Principles of Universal Design and their Guidelines

The Seven Principles of Universal Design, as identified by the Centre for Universal Design. These seven principles are as follows:

1. Equitable Use
2. Flexibility in Use
3. Simple and Intuitive Use
4. Perceptible Information
5. Tolerance for Error
6. Low Physical Effort
7. Size and Space for Approach and Use

These principles provide a checklist against which a final product can be evaluated, and a guide for a designer throughout the design process. The principles are further fleshed out by an abstract description, and a set of four of five guidelines, as provided below:

Principle 1: *Equitable Use*

The design is useful and marketable to people with diverse abilities.

A. Provide the same means of use for all users: identical whenever possible; equivalent when not.
B. Avoid segregating or stigmatizing any users.
C. Make provisions for privacy, security, and safety equally available to all users.
D. Make the design appealing to all users.

Principle 2. *Flexibility in Use*

The design accommodates a wide range of individual preferences and abilities.

A. Provide choice in methods of use.
B. Accommodate right- or left-handed access and use.
C. Facilitate the user's accuracy and precision.
D. Provide adaptability to the user's pace.

Principle 3. *Simple and Intuitive Use*

Use of the design is easy to understand, regardless of the user's experience, knowledge, language skills, or current concentration level.

A. Eliminate unnecessary complexity.
B. Be consistent with user expectations and intuition.
C. Accommodate a wide range of literacy and language skills.
D. Arrange information consistent with its importance.
E. Provide effective prompting and feedback during and after task completion.

Principle 4. *Perceptible Information*

The design communicates necessary information effectively to the user, regardless of ambient conditions or the user's sensory abilities.

A. Use different modes (pictorial, verbal, tactile) for redundant presentation of essential information.
B. Maximize "legibility" of essential information.
C. Differentiate elements in ways that can be described (i.e., make it easy to give instructions or directions).
D. Provide compatibility with a variety of techniques or devices used by people with sensory limitations.

Principle 5. *Tolerance for Error*

The design minimizes hazards and the adverse consequences of accidental or unintended actions.

A. Arrange elements to minimize hazards and errors: most used elements, most accessible; hazardous elements eliminated, isolated, or shielded
B. Provide warnings of hazards and errors.
C. Provide fail safe features.
D. Discourage unconscious action in tasks that require vigilance.

Principle 6. *Low Physical Effort*

The design can be used efficiently and comfortably and with a minimum of fatigue.

A. Allow user to maintain a neutral body position.
B. Use reasonable operating forces.
C. Minimize repetitive actions.
D. Minimize sustained physical effort.

Principle 7. *Size and Space for Approach and Use*

Appropriate size and space is provided for approach, reach, manipulation, and use regardless of user's body size, posture, or mobility.

A. Provide a clear line of sight to important elements for any seated or standing user.
B. Make reach to all components comfortable for any seated or standing user.
C. Accommodate variations in hand and grip size.
D. Provide adequate space for the use of assistive devices or personal assistance.