2018-10

The Semantic Student: Using Knowledge Modeling Activities to Enhance Enquiry-Based Group Learning in Engineering Education

Paul Stacey
Institute of Technology, Blanchardstown, paul.stacey@itb.ie

Transforming our World Through Design, Diversity and Education
G. Craddock et al. (Eds.)

doi:10.3233/978-1-61499-923-2-431

The Semantic Student: Using Knowledge Modeling Activities to Enhance Enquiry-Based Group Learning in Engineering Education

Paul STACEY[a,1]
[a] *School of Informatics & Engineering, Institute of Technology Blanchardstown, Dublin, Ireland*

Abstract. This paper argues that training engineering students in basic knowledge modeling techniques, using linked data principles, and semantic Web tools - within an enquiry-based group learning environment - enables them to enhance their domain knowledge, and their meta-cognitive skills. Knowledge modeling skills are in keeping with the principles of Universal Design for instruction. Learners are empowered with the regulation of cognition as they become more aware of their own development. This *semantic student* approach was trialed with a group of 3rd year Computer Engineering Students taking a module on computer architecture. An enquiry-based group learning activity was developed to help learners meet selected module learning outcomes. Learners were required to use semantic feature analysis and linked data principles to create a visual model of their knowledge structure. Results show that overall student attainment was increased when knowledge modeling activities were included as part of the learning process. A recommendation for practice to incorporate knowledge modeling as a learning strategy within an overall engineering curriculum framework is described. This can be achieved using semantic Web technologies such as semantic wikis and linked data tools.

Keywords. knowledge modeling, semantic Wikis, linked data, Engineering education

1. Introduction

1.1. Background

Learners undertaking the third year of a four year degree in Computer Engineering at the Institute of Technology Blanchardstown (ITB), Ireland, take twelve modules over 2 separate semesters for that year. Each module is weighted at 5 ECTS (European Credit Transfer System) credits. The author teaches two of these modules to these learners; *Software Design & Quality* (SD-Q), which is delivered in semester 1 of year 3, and

[1] Corresponding author, School of Informatics & Engineering, Department of Engineering, Institute of Technology Blanchardstown. Blanchardstown Road North, Dublin 15, Ireland; E-mail: paul.stacey@itb.ie.

Computer Architecture & Operating Systems (CA-OS), which is delivered in Semester 2 of year 3.

Over the course of two academic years, it was observed that the same learners were achieving higher grades in the module SD-Q (year 3, semester 1), compared with the module CA-OS (year 3, semester 2). This imbalance in learner attainment occurred over two consecutive academic years, 2015-16 and 2016-17 (the ITB academic year typically begins in September, ending in June. Semester 1 runs from September to January, and semester 2 runs from January to June). Paradoxically, learners would often comment that the SD-Q module was much more difficult, while the CA-OS module was perceived to be the less difficult of the two modules.

Both modules are delivered in similar teaching modes, by the same lecturer - the author. Both modules have the same assessment breakdown; 40% for course work, referred to as *continuous assessment* (CA), and 60% for a final written exam based assessment, referred to as *terminal exam*.

Notwithstanding the different content, and learning outcomes between the modules SD-Q and CA-OS, one key difference within the learning experience is the necessity for learners to engage in group-based conceptual modeling activities within SD-Q. Conceptual modeling skills – using, for example, the unified modeling language (UML) - are important for software designers & computer engineers, and thus it is a common learning outcome for learners taking courses in software design to obtain these skills.

Conceptual modeling within software design helps designers to understand the problem domain, before attempting to implement a software solution for the given problem. The object-oriented paradigm is built on this premise. Conceptual modeling is closely related to knowledge modeling, and seeks to visually represent the knowledge within a particular domain-of-enquiry. Conceptual modeling requires practitioners to engage in meaning-making, while exploring the concepts within a particular domain. Therefore, through the practice of conceptual modeling, a deeper level of understanding is acquired.

1.2. Hypothesis

It is the author's hypothesis that the task of conceptual modeling - inherent within the SD-Q curriculum - is enhancing the level of meaning making achieved by learners. The observed paradox of learners perceiving the SD-Q module to be more difficult than the CA-OS module, is a reflection of the fact learners are engaging in much deeper learning. Also, the lower learner attainment observed in the module CA-OS may be remedied by introducing conceptual modeling tasks within the CA-OS curriculum.

1.3. Objectives

The objectives of this study are to:

- introduce a conceptual modeling activity to the module CA-OS to enhance the level of learner meaning making within the learning experience.
- reduce the disparity in learner attainment between the modules SD-Q and CA-OS.
- assess the impact of conceptual modeling on learner engagement and attainment when applied to engineering learning environments & curricula not typically associated with conceptual modeling activities.

2. Meaning Making

Meaning-making within a learning context requires the learner to identify relationships between concepts within a domain-of-enquiry. Adopting a constructivist approach to teaching & learning requires the teacher to assess the level of meaning-making a learner has engaged in. Also, it is important for the learner to understand their own level of meaning-making. In many cases it is difficult to track and thus assess the stage of meaning-making for both the educator and the learner.

Making meaning is about making linkages. The more linkages between concepts, the higher the level of meaning making, or domain knowledge achieved. Conceptual models and mappings are a visual representation of concepts within a domain, and the relationships that exist between. Forming a relationship between two domain concepts is not an arbitrary task, and typically requires a good understanding of the domain.

There are many standards governing the representation of concept models. The Unified Modeling Language [18] is commonly used in software development. Typically a concept may be represented as a shape, such as a rectangle, with the concept labelled within the shape. Relationships or linkages between concepts are typically formed with a line drawn between concepts and a label or phrase that captures the nature of the relationship. Discovering and documenting relationships in a visual model requires the modeler to ask questions and develop a deeper knowledge of the subject which is the focus of the model. The conceptual model over time begins to visually document the knowledge available on the subject matter under investigation.

Visualization of knowledge represents a powerful tool to help develop the expert learner [13]. Ontologies are a powerful formalism of knowledge, and are a mechanism to capture a shared and common understanding of a domain which may then be communicated between people and software applications & systems [10]. Knowledge modeling is a significant activity to perform due to its complexity [11].

2.1. Knowledge Modeling in a Learning Context

The use of knowledge modeling as a learning tool has been growing in popularity. Chu et al. (2016) [9] note knowledge modeling as a practical tool for 21st century skills development. Bele and Rozman (2010) [7] show how semantic technologies can create a more immersive and engaging learning environment. Völkel (2011) [19] gives an in depth analysis of the benefits of personal knowledge models in reducing the limitations of cognitive processes, and highlights the usefulness of semantic technologies in providing a practical framework for personal knowledge management. Keßler, d'Aquin and Dietze (2013) [12] show the benefits of using Linked Data within educational environments.

Knowledge modeling skills are in keeping with the principles of Universal Design for Instruction (UDI) [16]. Learners are empowered with the regulation of cognition as they become more aware of their own development.

3. The Art of Knowledge Modelling: Tools of the Trade

3.1. Ontologies

An ontology is an explicit formal specification of the terminology that exists within a domain and the relationships between them [17]. Typically ontologies use classes to describe concepts in a domain. An ontology together with a set of individual instances of classes constitutes a knowledge base.

Ontologies are used to aid the automatic processing and sharing of knowledge. This implies they need to be machine readable. To be understood and processed by a computer, ontologies need to be formally defined and represented in a machine-readable format. Many languages have been devised to formalise ontologies. The Ontology Web Language (OWL) [4] language provides a way to formalise knowledge in a machine-readable format.

Semantic systems use ontologies to aid integration of heterogeneous data sets. Semantic systems seek to help exploit data and information within systems by enabling semantic search. Semantic search can uncover hidden knowledge. The Semantic Web [3] is an example of a semantic system. In the semantic Web, content is described in a meaningful way. Meaning is provided by ontologies.

Typically, the development of semantic systems is overly complex for casual users, as non-ontological expert users struggle with the formal logic of semantics [5]. However, there are many advanced tools to aid the development of semantic systems. For example, for ontology development Protégé is a commonly used tool. At a systems level Apache JENA [1] and Sesame [8] provide a rich framework of tools to help realise full semantic Web systems. Many frameworks will include reasoners such as the Pellet OWL-DL (Sirin et al. 2007).

Although the tools mentioned above abstract to some degree the complexity of ontology formalisms, there is still a significant learning curve for the novice user. More recently, more non-expert tools have become available to support the advent of the Semantic Web. These tools have evolved from the culture of WYSIWYG (what you see is what you get) Web authoring. Semantic Wikis are one example of authoring tools that have emerged to enable the development of Semantic Web based websites.

3.2. Semantic Wikis & Linked Data

Zaidan & Bax (2011) give a concise definition of semantic wikis:

> A semantic wiki is one that has an underlying knowledge model described on its pages. Classic or syntactic wikis are made up of text and untyped hyperlinks. Semantic Wikis, on the other hand, allow its users to identify information about the data described on the pages, and relations between pages, so that it may be inspected or exported as a database [20].

Semantic wikis are designed not only for collaborative use but can also be used for personal knowledge management [19].

The first implementation of semantic wikis began to appear in 2005. Since then, there have been numerous successful implementations. Currently Semantic MediaWiki [15] and OntoWiki [2] are the most popular semantic wiki implementations. Semantic

MediaWiki is a plugin for the successful MediaWiki software, whereas OntoWiki is a standalone semantic wiki engine. Both implementations are open source.

Semantic Wikis such as OntoWiki typically use a Linked Data [6] approach to formalising the underlying knowledge base. Linked Data is an approach for exposing, sharing and connecting structured data using Uniform Resource Identifier's (URI) and RDF [14]. The core principles of Linked Data provide the basic recipe for connecting data, information and concepts using Web technologies. Linked Data techniques use the generic graph-data model of RDF to structure and link data within a Linked Data approach. Linked data patterns are typically supported using RDF.

4. Study Overview

4.1. Methodology

Over a 4 week period, learners taking the CA-OS module were required to complete three knowledge modeling exercises related to their course of study. These exercises were completed after learners had completed 8 weeks of instruction of the CA-OS module. Two module learning outcomes were chosen as the target for the exercises and associated assessments, these are:

- Define the broad range of components that make up a Computer System.
- Describe how computer system components integrate to achieve higher levels of functionality and performance.

Extracts of the handouts provided to learners for the three exercises are provided in the appendix of this paper.

The first exercise (exercise 1) was referred to as Me_{now} while the final exercise (exercise 3) was referred to as the Me_{later} exercise. Learners also took an intermediate collaborative exercise (exercise 2) which was a bridging exercise between Me_{now} and Me_{later}. Prior to engaging in the three learning exercises, learners were given basic training (by way of a workshop) in knowledge modeling skills.

In the Me_{now} exercise, learners were given a list of 52 concepts related to their domain of enquiry (computer architecture & operating systems). Working individually, learners were asked to develop a personal knowledge graph by forming relationships between concepts, using a predefined set of labels. Where learners felt the labels did not capture the meaning of the relationship they were encouraged to define their own labels.

Learners were instructed not to perform any additional reading to enhance their knowledge during this exercise, but to capture their current personal understanding of the subject matter through the documentation of linkages. Learners were given a simple marking rubric which showed that assessment for this exercise was solely based on engagement and participation in a reflective practice; not the number of linkages contained within their personal knowledge graph. Learners visualized their personal knowledge graphs using the free online tool draw.io or Microsoft Visio. When completed, learners submitted their knowledge models via Moodle and completed a learning journal where they were asked to reflect on the exercise and their current learning related to CA-OS.

In exercise 2, learners engaged in a collaborative knowledge modeling exercise. In this exercise groups of learners were asked to research a particular topic related to CA-OS, and capture their findings in a collaborative knowledge model. Learners submitted

their collaborative knowledge model again through Moodle, and were asked to individually reflect on the exercise and their personal learning through a learning journal.

In the final knowledge modeling exercise (Me_{later}), learners were given a practical exercise to complete as a group. Following the exercise, the entire class were asked to develop an overall knowledge model of the entire CA-OS subject matter studied to date. Learner's shared knowledge structures were captured within a semantic wiki (using OntoWiki).

Once completed, learners were asked to revisit their original personal knowledge graph from exercise 1, and update it to capture their current level of meaning making. Learners submitted their Me_{later} personal knowledge model via Moodle, and again engaged in reflective practice through the completion of an online learning journal.

4.2. OntoWiki

To support exercise three (Me_{later}) detailed above, an instance of OntoWiki was installed on the author's personal Webserver[2]. Each learner was given a personal account and secure login details. A video tutorial was produced to guide learners through the process of collaborative knowledge modeling using the OntoWiki environment. A screen shot of the OntoWiki environment is shown in Figure 1 below.

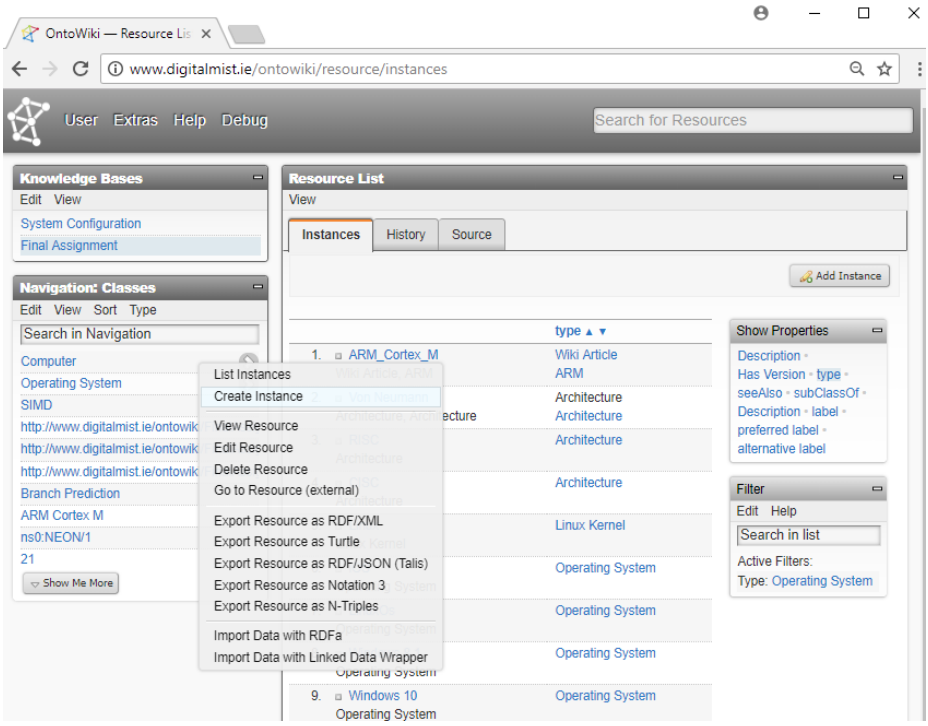

Figure 1. Screenshot of the OntoWiki installation used to support exercise 3.

[2]An instance of OntoWiki can be found at the following Web address http://www.digitalmist.ie/, and following the link OntoWiki (site) at the top of the page. For access to a free OntoWiki account please contact the author.

5. Results

Overall it was found that learners engaged well with all three exercises. Within each of the learning journals, learners noted the usefulness of having a clear view of their level of meaning making thus far, as this gave them confidence in what they do know and highlights areas where additional learning is needed. This self-regulation of cognition was evident within the knowledge maps produced by learners. Within exercise 1, all learners' personal knowledge graphs contained many missing linkages between several concepts. As learners progressed through the exercises it was observed that the updated knowledge graphs visualized learner's deeper meaning making, as all concepts contained linkages. Also, additional linkages and updated labelling was evident in all Me_{later} personal knowledge graphs.

5.1. Analysis

Analysis of learner attainment for the academic year 2017-18, compared to the previous two academic-years, showed that learners taking the Computer Architecture & Operating Systems (CA-OS) module scored similarly to the previous semester's module Software Design & Quality (SD-Q). In fact the average % grade was slightly higher for CA-OS for the academic year 2017-18. When compared to the academic years 2015-16 and 2016-17 there was a marked improvement in % points earned. For the years 2015-16 and 2016-17 the average % grade was 10-15% lower in CA-OS for those two years.

Comparison of the number of learners obtaining D and F grades (< 40%), shows that in the academic years 2015-16 and 2016-17, there was a marked difference in the percentage of learners obtaining Ds and F grades between both modules. On average, 20% more learners achieved these low grades in the CA-OS module as compared to SD-Q. In the year 2017-18 this differential was reduced to ~0; with learners obtaining comparable grades between both modules, and in line with SD-Q from the previous 2 years.

As the CA component to the CA-OS module has changed, it is important to explore whether there was an improvement in terminal exam grades attainment. For the years 2015-16 and 2016-17, learners on average attained 10-12% lower in the CA-OS terminal exam as compared with the SD-Q terminal exam for both years. In the 2017-18 academic year, with the introduction of knowledge modeling activities within the CA-OS module - the average differential between both module's terminal exam results was reduced to ~0%. Meaning learners performed equally well across both modules, and thus the discrepancy in achievement was no longer observed in either CA grades or terminal examination.

6. Discussion & Conclusion

The inclusion of knowledge modeling learning activities within an engineering learning environment has had an overall positive impact within the classroom, as well as learner attainment. However, tools such as OntoWiki to support collaborative semantic knowledge modeling environments are difficult to setup for use within the classroom. The installation of these tools is not for the novice user, and difficulties were encountered.

The learners who engaged in these exercises had prior knowledge of Web technologies as well as object modeling methodologies, which reduced the learning curve in terms of adoption within the classroom. The recommendation would be to incorporate knowledge modeling as a learning strategy within an overall curriculum framework and not solely within one module where learners do not have a basic understanding of object modeling and Web technologies.

It was notable that there was a significant barrier to the possibility of plagiarism within these activities, as learners were developing personal knowledge maps.

As Kinchin (2016) [13] notes, human beings are meaning makers and the goal of education is the construction of shared meaning. Semantic web technologies such as semantic wikis and linked data tools are ideal to help achieve this goal within engineering learning environments.

References

[1] Apache Jena (2010) A free and open source Java framework for building Semantic Web and Linked Data applications, [Online]. Available at http://Jena.Apache.Org (Accessed: 31 May 2018).

[2] Auer, S., Dietzold, S., & Riechert, T. (2006). OntoWiki–a tool for social, semantic collaboration. In International Semantic Web Conference (pp. 736-749). Springer, Berlin, Heidelberg.

[3] Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. Scientific american, 284(5), 34-43.

[4] Bechhofer, S. (2009). OWL: Web ontology language. In Encyclopedia of database systems (pp. 2008-2009). Springer US.

[5] Bernstein, A., & Kaufmann, E. (2006, November). GINO–a guided input natural language ontology editor. In International Semantic Web Conference (pp. 144-157). Springer, Berlin, Heidelberg.

[6] Bizer, C., Heath, T., & Berners-Lee, T. (2011). Linked data: the story so far. In Semantic services, interoperability and web applications: Emerging concepts (pp. 205-227). IGI Global.

[7] Bele, J. L., & Rozman, D. (2010). Semantic technologies in e-learning. M. Auer (ed.), J. Schreurs (ed.). Academic and corporate e-learning in a global context. Wien: International Association of Online Engineering, 431-439.

[8] Broekstra, J., Kampman, A. and Van Harmelen, F. (2002) Sesame: A generic architecture for storing and querying rdf and rdf schema. In International semantic web conference (pp. 54-68). Springer, Berlin, Heidelberg.

[9] Chu, S. K. W., Reynolds, R. B., Tavares, N. J., Notari, M., & Lee, C. W. Y. (2016). 21st century skills development through inquiry-based learning: From theory to practice. Springer.

[10] Davies, J., Fensel, D., & Van Harmelen, F. (Eds.). (2003). Towards the semantic web: ontology-driven knowledge management. John Wiley & Sons.

[11] Gaeta, M., Orciuoli, F., & Ritrovato, P. (2009). Advanced ontology management system for personalised e-Learning. Knowledge-Based Systems, 22(4), 292-301.

[12] Keßler, C., d'Aquin, M., & Dietze, S. (2013). Linked Data for science and education. Semantic Web, 4(1), 1-2.

[13] Kinchin, I. M. (2016). Visualising powerful knowledge to develop the expert learner: A knowledge structures perspective on teaching and learning at university. Springer.

[14] Klyne, G., & Carroll, J. J. (2006). Resource description framework (RDF): Concepts and abstract syntax.
[15] Krötzsch, M., Vrandečić, D., & Völkel, M. (2006, November). Semantic mediawiki. In International semantic web conference (pp. 935-942). Springer, Berlin, Heidelberg.

[16] Mcguire, J. M., Scott, S. S., & Shaw, S. F. (2006). Universal design and its applications in educational environments. Remedial and special education, 27(3), 166-175.

[17] Peirce, C. S. (1935). Collected papers. Scientific metaphysics (Ed. by C. Hartshorne and P. Weiss.), Vol. VI.

[18] UML, O. (2001). Unified modeling language. Object Management Group.

[19] Völkel, M. (2011). Personal knowledge models with semantic technologies. BoD–Books on Demand.

[20] Zaidan, F. H., & Bax, M. P. (2011). Semantic wikis and the collaborative construction of ontologies: a case study. JISTEM-Journal of Information Systems and Technology Management, 8(3), 539-554.

Appendix

Exercise 1: Me_{now} – Personal Knowledge Map (5%)

Purpose: To allow learners identify relationships between concepts and assess their own level of understanding.

Type: Individual Exercise **Method**: Knowledge formalism using data graphs.

Targeted Learning Outcomes:

- Define the broad range of components that make up a Computer System.
- Describe how computer system components integrate to achieve higher levels of functionality and performance.

Submission: Personal Knowledge Map & Journal Entry (Moodle)

Exercise Description:

You are required to create a visual model of your current knowledge of Computer Architecture & Operating Systems. Using the Computer Architecture & Operating Systems concepts in Appendix B, develop a Knowledge/Concept Map using the ***subject–predicate–object*** data graph described in Appendix A. you should attempt this exercise with minimal background reading of the concepts. Try it out and see what your current level of understanding is. As we have seen in ER and Object-Oriented modelling, it is very difficult to model something if you don't understand what it is!

Note the following:

- Each concept in Appendix B should be represented as a node in the graph using a circle (see Figure 1 below for an example). Literals are values that are not in themselves concepts. For example: Person – hasName – "John" ; John is a literal. A person (the **subject** of discussion, what we are talking about) has the name (**predicate**) John (the **object**, which is a literal, but could be another subject or node).

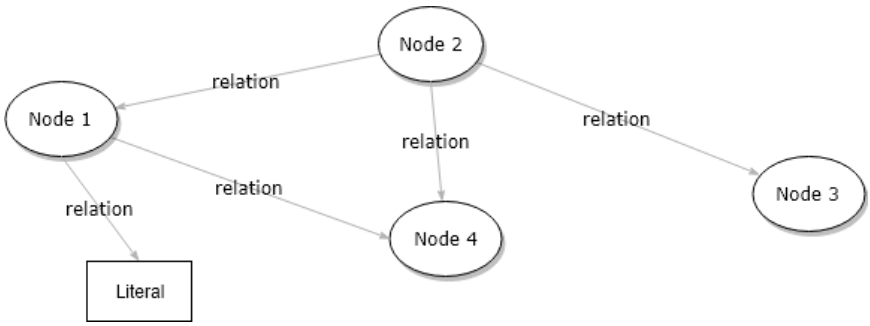

Figure 1 Concept Model

- Each relationship between the concepts in Appendix B should be one of the Relationship labels shown in Appendix A. If you feel a relationship does not exist create your own custom one, call it custom:<inset relation> (see Figure 2 below for an example):

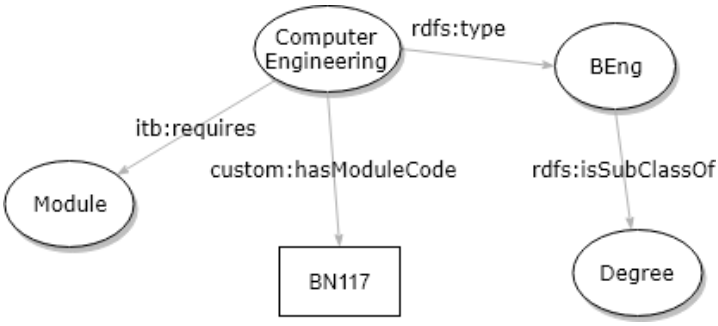

Figure 2 Computer Engineering Concept Model

Exercise 2: (10%) – Collaborative Knowledge Modelling

Purpose: Assess the level of shared-meaning making

Type: Group Exercise (~5 per group) **Method**: Knowledge formalism

Targeted Learning Outcomes:

- Define the broad range of components that make up a Computer System.
- Describe how computer system components integrate to achieve higher levels of functionality and performance.

Submission: Collaborative Knowledge Map and Journal Entry (Moodle)

Exercise Description:

The ARMv7 architecture introduced a number of substantial changes to its predecessors: *Thumb2*, *TrustZone*, *Jazelle-RCT*, *Neon*. There are a number of Cores designed by ARM holdings that implement the ARMv7A architecture. One of these cores is the ARM Cortex-A8. The ARM CortexA8 is a 32-bit processor core licensed by ARM Holdings implementing the ARMv7-A architecture. It was the first ARM processor to incorporate all of the new technologies available in the ARMv7 architecture. The processor features a high-performance, superscalar microarchitecture.

In this exercise you are to produce a knowledge/concept map (as you have for Exercise 1) detailing the features of the Cortex-A8's ARMv7A micro-architecture. You and your group will need to research implementations of this core/architecture and their use in products, specifically the "Beaglebone Black".

Your knowledge map should at a minimum model the following concepts:

The 3 main profiles of the ARMv7 Architecture

- The application “A” profile
- The Real-time “R” profile
- The Microcontroller “M” profile

Including concepts related to:

- Superscalar processors, pipelining
- Branch prediction
- NEON etc.

Additional Concepts

- ARM Cortex A8
- ARMv7
- Texas Instruments AM3358x Sitara Processors.
- Linux
- Beaglebone black.
- Raspberry Pi 1 Model B+

Exercise 3: Me_{later} – Semantic Wiki (10%)

Purpose: Document and share the level of shared-meaning making

Type: Class & individual

Method: Learners are required to capture knowledge structures within a Semantic Wiki using a Directed Acyclic Graph (DAG) concept graph using Linked Data and Semantic Web principles. The knowledge map should be serialised and validated, allowing queries to be answered against the knowledge model.

Targeted Learning Outcomes:

- Define the broad range of components that make up a Computer System.
- Describe how computer system components integrate to achieve higher levels of functionality and performance.

Submission: Semantic wiki contribution, personal knowledge map & journal entry (OnotoWiki & Moodle)

Exercise Description:

Having completed the BeagleBone development and analysis exercises on Moodle. Attempt the following:

The C code shown in Listing 1 below calculates a product matrix C by multiplying two matrices A & B.

- Implement the code in Listing 1 and produce an executable compatible with the Beaglebone black. Initially, build your application **without** support execution using the NEON processor.

- Next create an executable which will ensure the appropriate code will use the onboard NEON technology.

- Using Streamline technology (profile drilldown) analyse the performance benefits of using the NEON functionality. What you should find is the product matrix C can be executed much faster when built for NEON and subsequently executed.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
//#include <arm_neon.h>

int main(void){

          clock_t start, end;
          float elapsed_time;
          start = clock();

          float aa=1,bb=1,cc=1,dd=1;
          //float32x4_t ff={1,1,1,1};

          for( i=0;i<100000;i++){
                    aa=1,bb=1,cc=1,dd=1;
                    //ff=vdupq_n_f32(1);
                    for( j=0;j<100;j++){
                              aa=aa+aa;
                              bb=bb+bb;
                              cc=cc +cc;
                              dd=dd+dd;

                              //ff=vaddq_f32(ff,ff);
                              //printf("ff is = %f\n",ff[1]);
                    }
                    //printf("dd is = %f\n",dd);
                    //printf("ff is = %f\n",ff[1]);

          }

          printf("Answer = %f\n",dd);
          //printf("Answer = %f\n",ff[1]);

          end = clock();
          elapsed_time = (float)(end - start) / (float)CLOCKS_PER_SEC;
          printf("Elapsed time: %f seconds\n", elapsed_time);

          return 0;

}
```

Listing 1 NEON Experiment Code

```
    -mcpu=cortex-a8 -mfpu=neon -mfloat-abi=hard -ftree-vectorize -ftree-
vectorizer-verbose=1 -funroll-loops -mvectorize-with-neon-quad
```

Listing 2 Compiler Optimisation Flags