Global Journal of Engineering Education

Employing a DFD model to facilitate the management of final-year student projects in computer engineering

Basem Y. Alkazemi† & Grami Mohammad A. Grami‡

Umm Al-Qura University, Makkah, Kingdom of Saudi Arabia†
King Abdulaziz University, Jeddah, Kingdom of Saudi Arabia‡

ABSTRACT: In theory, small-scale projects are genuine opportunities to stimulate students' creativity and introduce them to the job market. Projects provide students with the chance to practise the knowledge they gained over years of education. However, the actual implementation of theory faces more practical challenges like successful and effective management. In this study, the authors identified a shortcoming in managing student projects, which could result in negative repercussions. The problem is the lack of carefully-designed, well-managed plans. This problem subsequently affects the execution of the project, students' productivity and further complicates managing their projects at later stages. The article reports on the experience of using Data Flow Diagram (DFD) to improve project management by dividing the work into clear-cut segments between members of the group. The authors, then, attempt to test and develop a set of guidelines to enable the implementation of this approach in an effective way. Results are generally positive and participants' projects were better defined and organised as a result. Project evaluators were equally impressed as they were able to identify the amount of work done and by whom.

Keywords: Data flow diagram, student projects' management, teaching computer engineering students

INTRODUCTION

Successful projects are usually the result of meticulous, well-designed and clearly defined plans [1]. Plans, therefore, are of significant value as they must clearly show the purpose and objectives of the project. They should also have a carefully designed framework, which accounts for the distribution of tasks between members of the group, expected duration and workload at various stages.

Planning and management skills, however, are not naturally inherited from the typical computer engineering curriculum. Students, therefore, should receive proper training prior to the commencement of their projects. One reason as to why the principles of project management are particularly tricky for computer engineering students may originate from the exciting scientific materials, which mainly centre on the technicality of building systems rather than managing and planning the stages of building the system. Despite previously studying *team organisation* and *project planning* in their respective software engineering course, the fact of the matter is that students hardly develop the real management skills required upon embarking on their respective projects later.

With this in mind, this article is based on an attempt to supervise five undergraduate students in Umm Al-Qura University (UQU), Makkah, doing two different projects over a period of six months. The authors also report on examining three more final year projects to obtain the BEng degree. They observed a recurring theme in students' inability to succeed in generating project plans, which is due to poor analysis of the system under development. Computer engineering students tend to build systems following a simplistic bottom up approach, usually by searching for needed components, buying them and connecting them on electronic boards hoping to achieve the required functionality and to meet supervisors' requirements.

Although this approach can be described as very basic at best and suitable only for very small projects, it still fails to account for any progress students make and to track any modification in later stages. Moreover, supervisors might find it equally difficult to monitor students' progress and check their attainment, as the work is not explicitly defined and organised. In other words, the lack of functional components throughout the analysis of systems can have a negative impact on students' ability to develop effective project plans. Hence, they lose track of their progression throughout the project. In addition, chaotic plans can also have an impact on supervisors in distributing marks over the activities accomplished in the project. Supervisors may not be able to evaluate accurately *who did what* and team collaborations as a result.

Having identified the shortcomings in students' management skills, the authors consequently propose a well-established concept described in software engineering courses - though not very popular in many institutions including these two - to be implemented in current approaches. The proposed utilisation of a data flow diagram (DFD) model helps analyse student projects more accurately from a system point of view (described below) [2]. Although this model might not be very popular in project management disciplines, it is still useful in systems development, where students will be able to analyse different components of systems at different levels of abstraction, and define the inputs and outputs of every component.

This article documents the experimentation with applying DFD at UQU to help students organise their project plans. It also reports on how DFD helped supervisors evaluate these projects. Additionally, the article presents a set of guidelines that are formulated by the concepts of DFD and *system model* as discussed in the literature review part of this article.

The organisation of the article is as follows: the concept of DFD and the different levels of abstractions are presented in the second section. An overview of a conceptual system model that establishes a context for analysing the systems to be developed is provided in the next section. The proposed guidelines are given in the following section. The methodological framework of this work is described in the fifth section. In the penultimate section, the most significant findings are discussed. Finally, the conclusions and potential future work are stated in the last part of the article.

DATA FLOW DIAGRAM (DFD)

Data flow diagram is an analysis technique used to model a system from the perspective of data flowing throughout the system itself. The literature suggests that DFD can be used to modernise old legacy systems and is equally important in *requirement elicitation*. In software engineering, DFDs can be used to model the structure of software systems [3-6]. The authors assume that DFD could help produce considerably more accurate project plans. The DFD model by default facilitates organising student projects in a logical manner, as well as developing their understanding of the problem they are about to tackle. Furthermore, this approach covers most of the analytical skills expected from computer engineering graduate students. The syntax of the DFD model depicted in Figure 1 is used to analyse the insulin pump control system.

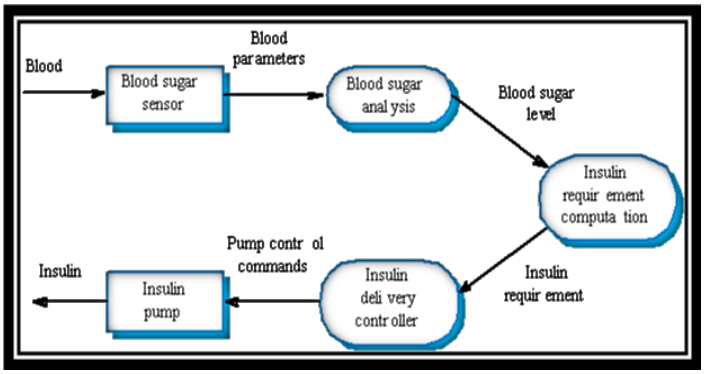

Figure 1: DFD for insulin pump control system.

DFD can function as a modelling technique capable of analysing customer requirements and transforming them into a visible structure. It is mainly used in software development to analyse functional requirements and establish the full design of the system under development. So, different components of the system can be identified at various level of abstraction. This can help to estimate the project duration more accurately as it identifies the entire functional component, hence, make it easy to estimate the task for each one.

By default, DFDs help users understand the various components of a system being developed and how to organise the component within that system. Their main objective is to track the flow of data once entered into a system up to the point of completing the project, having established the processing of data by the system and the production of an output. DFD is composed of various levels, each elaborates on the level superseding it such as:

- Level 0: this level is concerned with identifying the source and sink of data that describes the inputs and outputs of a system at its entirety.
- Level 1: this level identifies the basic functional elements of a system and the sequence of their interaction.
- Level 2: identify the sub-components of each functional element identified in 1 and their organisation.
- Level n: identify the building blocks of sub-components defined at Level n-1.

Data considered in software development could be any readable text, however, in computer building data will be a set of 0's and 1's only where 0 represents low voltage and 1 represents high voltage potential. So, monitoring the flow of the input data throughout a system and how it is going to be manipulated are key factors to consider at designing and building a system according to the DFD model. The complexity of the DFD model is omitted as it is more relevant to software than hardware systems. However, the basic principles of the DFD approach are useful in hardware design as it helps students to analyse the various parts of their system.

SYSTEM MODEL

The main objective of students' projects is to build a functional system. Here, the authors introduce the concept of a *system* in simple terms, which in conjunction with the previous section should help readers appreciate the need for an efficient management procedure.

A common model of a system can be described in abstract terms as intricate because of the complicated nature of different components [7]. The components are the parts used to build the system. So, a system is a composition of a set of components. Each component is composed of sub-components and sub-components are composed of sub-sub-components. This process of decomposition should continue until reaching to an atomic component that cannot be decomposed any further. For example, a *Multiplexer* is a trivial system that its internal structure is described in Figure 2 below.

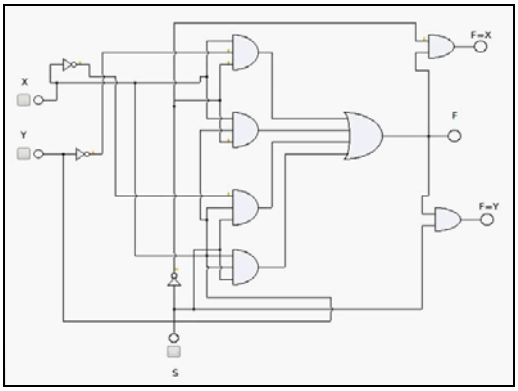

Figure 2: Internal structure of a Multiplexer.

It is composed of six AND gates, one OR gate, and three inverters. Each gate, in turn, is composed of a number of bipolar junction transistors (BJT) and resistors. For example, the AND gate is one level of abstraction that can be considered as a system. Its internal components are composed of two BJT transistors and three variable resistors as illustrated in Figure 3.

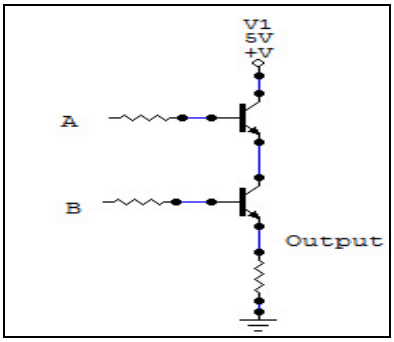

Figure 3: Internal structure of the AND gate.

Theoretically speaking, the system establishes the interaction between components, provides a way to control their execution and supplies components with the necessary inputs to work. Components are responsible for providing a system's functionality. The components of a system interact with each other through standard interfaces that define a set of characteristics required by a system and exhibited by components. If characteristics of the interference are not matched, then, components will not fit into a system building. The authors identified the different levels of abstraction for a system in the computer systems as described in Figure 4.

Level 4	Computer Networks	Communications
Level3	Applications	Functionality
Level 2	Operating System	Control
Level 1	Digital Circuits	Abstraction
Level 0	Electronic Circuits	Building Block

Figure 4: Different levels of abstractions in computer systems.

Starting at Level 0, every computer system is composed of a number of integrated circuits (ICs) that are designed to build a real working system. This level can be mapped to the elementary courses in computer engineering, where students learn various types of electronic components in a number of modules, such as Circuit Analysis 1 and 2,

electronic devices and digital electronics. In other words, Level 0 is the building block to any student who wishes to graduate from a computer engineering department.

Level 1 expresses how the electronic circuits are configured to provide different abstractions for higher level components. For example, a NAND gate is an abstraction of a composite of two transistors connected in series. Therefore, the second level that computer engineering students should learn is how to generate abstractions from electronic circuits. Also, this level concerns with teaching students the necessary skills to organise digital circuits in order to build a basic computer system. A number of modules should be given to students at this level, such as digital design and computer architecture.

Level 2 establishes control over the digital circuits designed and implemented at Level 1. At this level, hardware can interface with software. During this level, students learn machine languages and operating system design in order to facilitate the interfacing between their hardware components (i.e. the digital circuits) and any software system that can be used in future to utilise these circuits.

Level 3 adds functionality to the hardware brought forward from Level 2. Students start learning some programming skills in order to develop their own applications utilising the hardware systems they have built when they studied digital design and computer architecture modules.

So far, the authors have a self-contained computer system that can read requests, process and manipulate data, and return results to users. However, it is not always possible to have every type of application in a single computer system. There is a need to interact with other systems in order to perform more functionality. Therefore, Level 4 is meant to facilitate communication between computer systems. Students at this level learn the various aspects about building LANs, WANs and their configurations.

THE PROPOSED GUIDELINES

Based on the system model and the levels described by the DFD model, the authors drafted a set of guidelines that computer engineering students can follow to develop more accurate project plans. There are five stages that need to be considered when planning and designing systems as follows:

- Identify the major inputs and outputs of the system to be built. This step describes what should be provided to a system in order to start executing and what is expected to be provided based on that input.
- Identify the various functional components that comprise a system at a high-level of abstraction. In this step, students need to think abstractly about the different functionalities that should be incorporated into a system in order to provide its expected output. The total functionality of a system is the sum of the sub functionalities provided by components. This can be represented by the following equation: $F = \Sigma$ fn, where F is the total functionality provided by the system, and f is the distinct functionality provided by components. So, the total functionality of a Full Adder system can be counted as F = fXOR1 + fXOR2 + fAND1 + fAND2.
- Identify the sequence by which functionality is going to be organised. The flow of data from the input to the output of a system passes through the various components that comprise that system. The correct sequence of data flow significantly affects system's functionality as incorrect sequence might result in providing erroneous output. This activity concerns prioritising the functionality that should be built first in the system in order to feed other components with the correct input values. An analogy to this idea can be seen in the building of a living room. Doors and windows cannot be fixed unless the room is built.
- Identify the interface of each component with others in the system to be built. After identifying the various functional components abstractly in the previous step, it is appropriate now to think in depth about the possible components that can fill in the functionality identified earlier. Every component must be analysed in terms of its required input and provided output. The required input represents the data that must be provided to a component in order to start executing. For example, an IC must be connected to a 5 volts source with 1mA current and supplied with data value: 101010. The provided output is the set of data values that result after components' execution.
- Identify the interaction protocols between components. In this step, the characteristics of the channels that connect components in a system are examined. A channel could be a simple set of wires or a more complex data bus with specific bandwidth specification.

Then, students should return to the first step to start iteration and treat the identified components as systems if possible. Some large systems are built from components that cannot be obtained as a single piece but they need to be constructed from a number of sub-components. As a result, this step proposes iterating over the above steps in order to build the component that is going to be part of the system under development.

METHODOLOGY AND EVALUATION

Applying the approach described earlier in project development allows successful utilisation of several tools, such as the Gantt Chart [8] and MS Projects [9] to track students' progress in the project. One of the projects supervised by the authors was concerned with designing a prototype of a digital chip to control the operation of traffic lights in case of

emergencies. The project was carried out by two students over a period of three months, including the writing up of the final report. Live GNOME Planner [10] was used to manage progress as it is a general purpose project management tool that is free and open-source software [11]. The students were requested to update their progress on that chart regularly at the end of every working day. They were also requested to use a simulation software product of their choice to build their design. For this project they used NI *Multisim* [12].

The first step in designing the required chip was to ask the students to define abstractly the input and output of the digital chip they were going to build. So, they identified that the input would be an infrared signal and the output would be in the form of an electric pulse that varies from 0 to 1. If the output is 0, then, the chip is in standby state and if the output is 1, then, the chip is in control of the operation to interrupt the traffic light system's operation.

The next level was to identify the functionality that would be performed inside that chip in order to interpret the coming signal and provide the output according to the required specification. The students analysed the flow of data from the input to the output of the chip and identified the main functionality of the components, and placed them in the following sequence as: signal detecting, interpreting, validating, signal firing, decision making, state storing, interrupting and state restoring. The students were asked to define the possible interfaces for each of the functional components they proposed and explain how they would interact with each other.

RESULTS AND DISCUSSION

The advantages of applying this approach can be observed in how it facilitates the generation of an accurate Gantt chart for monitoring students' progress and estimates accurately the amount of work required to finish their project. The project was split into 52 working units identified according to the effort needed to analyse the system requirement, generate design diagrams, buy or build the various components of the system, in addition to the report writing. The estimation of the size of a single unit is that it needs eight working hours. Every activity includes several working units and it has an owner who is responsible for executing it. Figure 5 illustrates a partial list of the activities identified in this project.

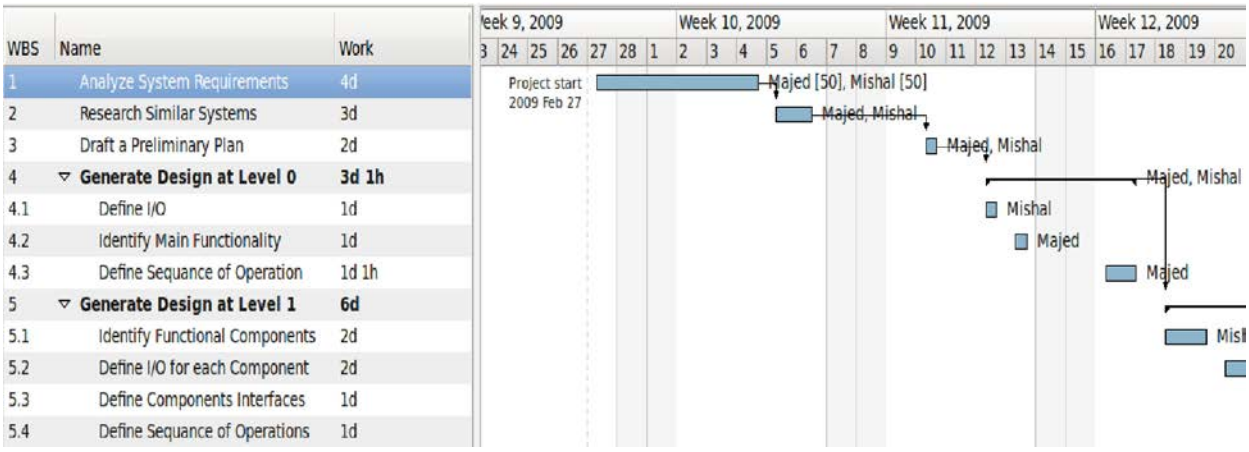

Figure 5: Partial list of activities on planner.

With respect to potential risks, risk management, such as sickness and faulty components were not considered in detail as the project was small in size. However, other factors that can have an impact include public holidays and late delivery of components by vendors. Despite that, the project was finished on time as planned and the students managed to present their work and attend the graduation ceremony as expected.

The findings of the study seem to confirm those of earlier studies in the sense that students taking part in final-year projects are constructively putting their knowledge into practice [13][14]. However, as the authors have witnessed, there should be more intervention from the supervisors' point of view to ensure that students' projects are conducted systematically. In fact, embedding the concept of DFD certainly helped organise the projects in ways usually unattainable if only traditional methods are observed.

Having discovered that DFD provides much-needed data for students in the form of the amount of work needed and the time-frame to accomplish different stages of the projects, the authors would draw on their own experience with students and argue that the process went smoother and faster than it would be possible without consulting DFD.

As a result, it would be advisable to at least consider implementing one form of data organising scheme, DFD or any similar alternative, when students embark on their final-year projects. This step cannot be achieved by students alone. In fact, even more experienced instructors/advisors/supervisors may require some orientation beforehand. However, despite the additional amount of training and effort to use this scheme, the authors believe the subsequent results are worthy of the extra time and effort.

CONCLUSIONS

This article reports on a new approach that facilitates managing student projects and helps supervisors to establish the necessary framework for managing projects in computer engineering, based on utilising the concept of a DFD model. A number of potential advantages were observed from applying the proposed approach to students' projects. These included facilitating the distribution of tasks over a team of students, identifying the ownership of activities, accurately measuring the extent of work needed, estimating the cost and duration of the project, tracking students' progress closely, facilitating error tracking and troubleshooting and, last but not least, increasing students' performance and productivity.

The design of the proposed approach is not yet finalised as the authors anticipate that further research on larger-scale projects is needed. However, as it stands, this article shows promising results of the attempt to implement an alternative approach to project management. The application of DFD to the research sample was greatly beneficial to students and supervisors alike and the authors expect that similar outcomes in different contexts would also be reported. It is worth mentioning that this work assessed the proposed approach by conducting a number of experiments to examine how DFD can really help students' productivity and supervisors' activities.

Publicising this approach on a wider scale and generating results based on the feedback obtained will be done at the next stage of this work.

REFERENCES

1. Schwalbe, K., *Information Technology Project Management*. (6th Edn), Cengage Learning (2010).
2. Roth, D.W., *Systems Analysis & Design*. (3rd Edn), Wiley Higher Education (2005).
3. Romney, M.B. and Steinbart, P.J., *Accounting Information Systems*. Prentice Hall (2008).
4. Alabiso, B., Transformation of data flow analysis models to object oriented design. *ACM SIGPLAN Notices - Special Issue: OOPSLA 88 Conf.*, 335-354 (1988).
5. Jilanic, A.A., Nadeem, A., Kim, T. and Cho, E., Formal Representations of the Data Flow Diagram: A Survey. Advanced Software Engineering and Its Application (ASEA), 153-158 (2008).
6. Adler, M., An algebra for data flow diagram process decomposition. *IEEE Transactions of Software Engng*, 14, **2**, 169-183 (1988).
7. Alkazemi, B.Y., A precise characterization of software component interfaces. *J. of Software***,** 6, **3,** 349-365 (2011).
8. Gantt, H. L., *Work, Wages and Profit*. The Engineering Magazine, Pennsylvania: Hive Publishing Company (1974).
9. Balmer, S., Microsoft Office Project Conference (2007), 7 April 2012, www.microsoft.com/Presspass/exec/steve/2007/10-30OPConferenceBallmer.mspx.
10. Live GNOME Planner Website (2005), 4 April 2012, http://live.gnome.org/Planner/Windows
11. Muffatto, M., *Open Source: A Multidisciplinary Approach*. Imperial College Press (2006).
12. National Instruments Website (2010), 7 April 2012, www.ni.com/multisim
13. Rojas, M.D., McGill, T. and Depickere, A., Project management in student information technology projects. *Inter. J. of Infor. and Communication Technol. Educ.*, 2, **4**, 24-38, (2008).
14. Grundy, J.C., A comparative analysis of design principles for project-based IT courses, *Proc. 2nd Australasian Conf. on Computer Science Educ. (ACSE),* 170-177 (1997).

BIOGRAPHIES

Dr Basem Alkezemi is currently holding the position of Vice-Dean of Umm Al-Qura University's IT Deanship for E-Government. He received his Bachelor degree in Electrical and Computer Engineering in 1999. He then went to study his MSc and PhD in Software Engineering at Newcastle University in the UK, which he received in 2004 and 2009, respectively. Dr Alkazemi has published a number of articles in regional and international journals and participated in many specialised conferences around the world. His main research interests are in software engineering, wireless sensor networks, computer supported education and e-government.

Grami M. Grami graduated in 2001 with a degree in English Language and Literature from King Abdulaziz University, Jeddah, in the Kingdom of Saudi Arabia. In 2003, he went to Essex University to study for his Masters degree, before embarking on a PhD project at Newcastle University, accomplished by 2010. He currently teaches English and Applied Linguistics at King Abdulaziz University. Grami has published a number of articles in international journals about IT and education, and is a current member of the editorial board of the Information Technology and Teacher Education Journal.