NATIONAL CERTIFICATES (VOCATIONAL)

SUBJECT GUIDELINES

INTRODUCTION TO SYSTEMS DEVELOPMENT

NQF LEVEL 2

December 2007

INTRODUCTION

A. What is Introduction to Systems Development?

Introduction to Systems Development provides the student with an introduction to and creates an awareness of the concepts relating to systems development. It introduces the principles of problem solving application using a computer programming language.

B. Why is Introduction to Systems Development important in the Information Technology and Computer Science programme?

It provides the student with the necessary foundation needed to understand the development of computer programs and the related technologies and processes involved.

C. The link between the Introduction to Systems Development Learning Outcomes and the Critical and Developmental Outcomes

The student will be able to identify and solve systems development problems faced by the business organisation by collecting, organising, analysing and critically evaluating relevant information. The student will recognise the interrelatedness of business organisational systems and problem-solving contexts.

D. Factors that contribute to achieving the Introduction to Systems Development Learning Outcomes

- Analytical and logical ability
- Keen powers of observation
- Ability to transfer skills from familiar to unfamiliar situations
- Meticulous nature
- Interest in computers and related topics

INTRODUCTION TO SYSTEMS DEVELOPMENT – LEVEL 2

CONTENTS

1 DURATION AND TUITION TIME

This is a one-year instructional programme comprising 200 teaching and learning hours. The subject may be offered on a part-time basis provided the student meets all the assessment requirements.

Students with special education needs (LSEN) must be catered for in a way that eliminates barriers to learning.

2 SUBJECT LEVEL FOCUS

The student will be able to explain systems development concepts.

3 ASSESSMENT REQUIREMENTS

3.1 Internal assessment (50 percent)

3.1.1 Theoretical component

The theoretical component forms 60 percent of the internal assessment mark.

Internal assessment of the theoretical component in Introduction to Systems Development Level 2 takes the form of observation, class questions, group work, informal group competitions with rewards, individual discussions with students, class, topic and semester tests and internal examinations. Lecturers can observe students when marking exercises from the previous day and asking class questions.

Assignments, case studies and tests can be completed at the end of a topic. Tests and internal examinations must form part of the internal assessment.

3.1.2 Practical component

The practical component forms 40 percent of the internal assessment mark.

Practical components include applications and exercises. All practical components must be indicated in a Portfolio of Evidence (PoE).

Internal assessment of the practical component in Introduction to Systems Development Level 2 takes the form of assignments, practical exercises, case studies and practical examinations in a simulated business environment.

Students may complete practical exercises daily. Assignments and case studies can be completed at the end of a topic. Practical examinations can form part of internal practical assessment.

- **Some examples of practical assessments include, but are not limited to:**
  - A. Presentations (lectures, demonstrations, group discussions and activities, practical work, observation, role-play, independent activity, synthesis and evaluation)
  - B. Exhibitions by students
  - C. Visits undertaken by students based on a structured assignment task
  - D. Research
  - E. Task performance in a “Structured Environment”

- **Definition of the term “Structured Environment”**

For the purposes of assessment, “Structured Environment” refers to a simulated workplace or workshop environment. It is advised that a practicum room is available on each campus for practical assessment.

- **Evidence in practical assessments**

All evidence pertaining to evaluation of practical work must be reflected in the students’ Portfolio of Evidence (PoE). The tools and instruments constructed and used to conduct these assessments must be clear from the evidence contained in the Portfolio of Evidence (PoE).

3.1.3 Processing of internal assessment mark for the year

A year mark out of 100 is calculated by adding the marks of the theoretical component (60 percent) and the practical component (40 percent) of the internal continuous assessment (ICASS).

3.1.4 Moderation of internal assessment mark

Internal assessment is subjected to internal and external moderation procedures as set out in the *National Examinations Policy for FET College Programmes*.

3.2 External assessment (50 percent)

A National Examination is conducted annually in October or November by means of a paper(s) set and moderated externally. A practical component will also be assessed.

External assessment details and procedures are set out in the *Assessment Guidelines: Introduction to Systems Development* (Level 2).

4 WEIGHTED VALUES OF TOPICS

TOPICS	WEIGHTED VALUE
1. Basic Concepts of Systems and Application Software	10
2. Software Development and Programming Languages Concepts	10
3. Computer Data Storage	10
4. Basic Computer Programming	60
5. Principles of Computer Program Quality Assurance and Project Viability	10
TOTAL	**100**

5 CALCULATION OF FINAL MARK

Internal assessment mark: Student's mark/100 x 50 = a mark out of 50 (a)

Examination mark: Student's mark/100 x 50 = a mark out of 50 (b)

Final mark: (a) + (b) = a mark out of 100

All marks are systematically processed and accurately recorded to be available as hard copy evidence for, amongst others, reporting, moderation and verification purposes.

6 PASS REQUIREMENTS

The student must obtain at least fifty (50) percent in ICASS and fifty (50) percent in the examination.

7 SUBJECT AND LEARNING OUTCOMES

On completion of Introduction to Systems Development Level 2, the student should have covered the following topics:

Topic 1: Basic Concepts of Systems and Application Software
Topic 2: Software Development and Programming Languages Concepts
Topic 3: Computer Data Storage
Topic 4: Basic Computer Programming
Topic 5: Principles of Computer Program Quality Assurance and Project Viability

7.1 Topic 1: Basic Concepts of Systems and Application Software

Subject Outcome 1: Explain what software is and categorise the types of software.

Learning Outcomes:

The student should be able to:

- Explain the term software.
- Differentiate between the types of software and their purposes.
- Differentiate between proprietary and open source software.
- Outline the reasons for different versions within the same software.

Subject Outcome 2: Describe some features common to all types of application software.

Learning Outcomes:

The student should be able to:

- Identify and demonstrate the different features common to all types of application software.
- List and describe different types of application software and their use.
- Explain the purpose and use of the types of features common to all types of application software.
- Outline the processes for installing application software.

Subject Outcome 3: Define system software.

Learning Outcomes:

The student should be able to:

- Briefly describe the term system software.
- Define the operating system in terms of the tasks it performs in a computer.
- Define utility programs in terms of their use.
- Define language translators in terms of their purpose, with examples.

Subject Outcome 4: Name and describe microcomputer operating systems and operating environments.

Learning Outcomes:

The student should be able to:

- Name and describe different operating systems.
- Describe the environment in which the operating system operates.
- Outline the history of the different operating systems in terms of proprietary and open source.

7.2 Topic 2: Software Development and Programming Languages Concepts

Subject Outcome 1: Describe the generations of programming languages

Learning Outcomes:

The student should be able to:

- List the generations of programming languages in which they have evolved.
- Outline the types and levels of programming languages in terms of technicality, flexibility, user-friendliness and speed.
- Compare the strengths and limitations of programming languages.

Subject Outcome 2: Describe the uses for some of the most popular high-level programming languages.

Learning Outcomes:

The student should be able to:

- List and describe the most popular high-level programming languages.
- Explain the uses of high-level programming languages.
- Compare the advantages and disadvantages of high-level programming languages.

Subject Outcome 3: Describe concepts relating to object-oriented and visual programming.

Learning Outcomes:

The student should be able to:

- Describe object-oriented and visual programming methodologies with reference to EDP (Event Driven Programming) and RAD (Rapid Application Development).
- Explain the term visual programming language in terms of its concepts.
- Explain object-oriented programming languages in terms of the concepts involved.
  Range: Classes, objects, encapsulation, abstraction, inheritance, polymorphism

- Explain the relation between Visual programming, Rapid Application Development, Object Orientation and Object Oriented Programming.
- Explain object oriented programming in terms of the re-use of classes and the implementation of objects.
- List examples of object-oriented programming languages.

Subject Outcome 4: Name and discuss basic steps in developing a computer program.

Learning Outcomes:

The student should be able to:

- Name the basic steps for developing a computer program.
- Discuss briefly the basic steps involved in a computer program development cycle (PLDC).

Subject Outcome 5: Describe software development tools.

Learning Outcomes:

The student should be able to:

- Name examples of software development tools.
- Briefly describe these software development tools.

7.3 Topic 3: Computer Data Storage

Subject Outcome 1: Demonstrate an understanding of computer data types.

Learning Outcomes:

The student should be able to:

- Distinguish between data types and their examples.
- Distinguish categories of coding systems and their uses in a business environment.
- Explain and illustrate how data manipulation operations are performed on data types.

7.4 Topic 4: Basic Computer Programming

Subject Outcome 1: Describe the term problem solving.

Learning Outcomes:

The student should be able to:

- Name and describe the steps and techniques of problem solving
  Range: Understand the problem, devise a plan, carry out the plan, review the solution, apply changes, test and implement the plan - theoretical problem solving without referring to a specific programming language.

Subject Outcome 2: Produce and document an algorithm.

Learning Outcomes:

The student should be able to:

- Define an algorithm and explain what it is used for.
- Identify inputs, processes and outputs needed to construct an algorithm.
- Draw an IPO chart for a given problem.
- Identify the algorithmic structures needed to produce a feasible algorithm to solve a given problem.

Subject Outcome 3: Produce and document pseudo-code for a given problem.

Learning Outcomes:

The student should be able to:

- Define pseudo-code and explain what it is used for.
- Differentiate between an algorithm and pseudo-code with reference to the level of detail involved.
- Produce pseudo-code to solve a given problem by implementing logically-correct program and problem solving constructs and techniques.

Subject Outcome 4: Produce and implement alternate design methods to document a specification or solution for a given problem.

Learning Outcomes:

The student should be able to:

- List and briefly explain alternate methods for documenting and specifying a solution for a given problem.
- Use alternate methods to document a solution.

Subject Outcome 5: Implement a programming language to solve a given problem.

Learning Outcomes:

The student should be able to:

- Use an appropriate programming language for implementing the designed solution.
- Use the IDE (Integrated Development Environment) of the programming language to write the source code according to the solution designed.
- Compile and debug the developed program for syntax and logical errors.
- Test the correctness of the program using sample data.

7.5 Topic 5: Principles of Computer Program Quality Assurance and Project Viability

Subject Outcome 1: Describe the basic principles of program quality assurance.

Learning Outcomes:

The student should be able to:

- Explain the principles of good program documentation.
- Explain the principles of programming quality assurance (QA).
- Distinguish between validation and verification.

Subject Outcome 2: Describe the principles used to determine project viability

Learning Outcomes:

The student should be able to:

- Explain how to evaluate the viability of developing computer programs to solve problems.
- Identify the issues involved in assessing the viability of developing computer programs in terms of its design.

8 RESOURCE NEEDS FOR THE TEACHING OF INTRODUCTION TO SYSTEMS DEVELOPMENT – LEVEL 2

8.1 Physical resources

The following teaching aids should be made available, if possible:

- Lecture room
- Educator's computer connected to the Internet
- Data projector
- Electronic white board
- Networked computer room or library with Internet access and referencing software, for example Encarta Encyclopaedia
- Networked laser printer per five computers.

8.2 Human resources

- The lecturer must have completed an Information Technology-related subject at NQF Level 5 and be trained in outcomes-based education.
- It will an advantage if the lecturer has already been declared competent as assessor and/or moderator.

8.3 Other resources

- File per student for the Portfolio of Evidence (PoE) and 30 printing sheets of paper per student
- Computer-related books for referencing purposes