Module: Software Engineering 371

Module name:	Software Engineering 371
Code:	SEN371
NQF level:	7
Type:	Core – Bachelor of Information Technology
Contact Time:	66 hours
Structured time:	10 hours
Self-directed time:	84 hours
Notional hours:	160 hours
Credits:	16
Prerequisites:	SAD371, PRG272, PMM371

Purpose

In this module the student is expected to integrate all knowledge concerning programming into a functional project, demonstrating the analysis, design, planning, implementation, platforms, database development, distributed applications, and programming abilities. The requirements of the project will change and be adjusted throughout the course. In some situations the approach or choice of technology to be used will be dictated.

Outcomes

Upon successful completion this module, the student will be able to:

- Apply prerequisite knowledge in programming, database development, analysis and design, integrating these diverse components optimally to construct a complete, real-world system.
- Master design, engineering and implementation principles through practical implementation and defence of decisions made, demonstrating consideration of alternatives and sound evaluation criteria to make judgements.
- Produce the various types of document artefacts expected of a software engineer, communicating information effectively to various target audiences. Similarly, deliver such information through effective presentations.
- Manage a demanding and escalating workload, honouring deadlines.
- Verbalise knowledge correctly and succinctly in reports and tests.
- Function as individuals, in teams and in mentor-mentee relationships.
- Embrace responsibility for self and for advancing the success of peers.

Assessment

Assessment is performed using a variety of instruments:

- The *bona FIDE* (frequent, immediate, discriminating, empathic) philosophy is adopted. Formative tests are written on the last day of each of the first seven weeks. Assessed tests are handed back on the first day of the following week and discussed. Any trends identified as deficient across the group are addressed through ad hoc presentations created for the specific purpose required. During the final week, a summative test is written in the middle of that week to enable announcement of the final results on the last day of this module.

- Practical work is evaluated using appropriate rubrics by a variety of assessors, including domain experts and peer evaluations. Students also evaluate their peers on how helpful they were to them with advice while evaluating their projects. Final project assessments are performed by a panel of accomplished software engineers.
- Throughout the duration of the module, current results are published. This includes projected final results based on current achievement, to inform students' expectations and spur increased effort timeously where required

Teaching and Learning

Learning materials

- Daily notes related to the required outcome for the day will provided to each student.
- Electronic copies of various artefacts will be provided to the students via an electronic communication channel.

Learning activities

The teaching and learning activities consist of a combination of formal lectures on theoretical concepts, exercises and discussions. These discussions are dialogue-oriented to stimulate peer discussion on practice. Several mandatory assignment and several smaller projects must be completed during the course. The experiences and progress on these practical components form the content of class discussions.

Notional learning hours

Activity	**Units**	**Contact Time**	**Structured Time**	**Self-Directed Time**
Lecture		52.0		22.0
Formative feedback		8.0		
Project	2	6.0		15.0
Assignment	3			9.0
Test	4		8.0	16.0
Exam	1		2.0	22.0
		66.0	**10.0**	**84.0**

Syllabus

- Software engineering; software development lifecycle; modelling, abstraction, standards, and specification; systems and architecture; layering and separation of concern; review and application of object-oriented principles; review of object oriented programming.
- Application layering; the data layer; databases; data topologies; relational databases; data normalisation (0NF, 1NF, 2NF, 3NF); distributed heterogeneous data stores; transactions; database integrity rules; logs and journals; coordinators and participants.
- The data access layer, data access technologies, data providers, dataset components; the user interface layer, graphical user interfaces, data binding in UI components, rich vs. thin clients, interfaces and channels.

- Business logic, business rules, inference engines, backward vs. forward chaining, workflow management, state machines, code-based vs. data-based specification, tight binding and loose coupling for reusability.
- Concurrency and consensus; thread-safe programming; distributed applications. Linear data structures (stacks, queues, lists).
- Non-linear data structures, associative arrays (dictionaries, priority queues).
- Ordered data structures (binary trees, heaps).
- Review architectural and design decisions.