A Descriptive Formalisation of Student Management System (SMS)

Dr. Okengwu U. A[1]
Department of Computer Science,
University of Port-Harcourt, Port-Harcourt.
Nigeria

Dr.Ejiofor C. I[2]
Department of Computer Science,
University of Port-Harcourt, Port-Harcourt.
Nigeria

***Abstract* - The formalisation of Student Management System (SMS), can invariably contribute to the success, profitability and customer-based approach of Educational Institutes. The use of formal specification creates a formal approach for specifying the underlying functions and properties of the system. This paper has attempted to give a formal description of the activities of a SMS Using Zed notation. The interaction within the system is visualized using Unified Modelling Language (UML) sequence diagrams.**

Keywords: Student Mangement System (SMS), Z-Notation, UML.

1 BACKGROUND OF STUDY

Software systems are system built on the premise of software, integrating within hardware components. These systems over time have become so complex in term of components integration and software usage. The reliability of this system is the bedrock of critical safety systems forcing developer to move toward formal methods to verify system correctness (Rushby, 2000).

Formal methods are methods based on mathematical notations (representations) and analysis of software. It includes formal specification, specification analysis, proof transformation (development) and program verification (Spivey, 1998). The branch of mathematics utilized in formal method is discrete mathematics and mathematical concept based on set theory, Cartesian products, logic and algebra (Spivey, 1992 and Spivey, 1998).

Formal proof provide a verification avenue in which the developer can verify correctness and spot out deficiencies in formally created methods (Romanovsky, 2013).

The utilisations of formal method create design and implement formal systems. A formal system is a well-defined system or logical calculus of abstract though and concept using formalised mathematical models and languages. Each formal system is tied to a set of primitive symbols which finitely construct a formal language from a set of axioms through formations and inferential rules (Hunter, 1971). The system thus consists of valid formulas built up through finite combinations of primitive's symbols. These symbols are combined to form axioms (statements) in accordance with stated rules (Hunter, 1971). These statements more formally be can expressed as a set of finite symbols that can be used for constructing formulas (Hunter, 1971).
A student information system (SIS), student management system, school administration software or student administration system is a management information system for education establishments to manage student data. Student information systems provide capabilities for registering students in courses, documenting grading, transcripts, results of student tests and other assessment scores, building student schedules, tracking student attendance, and managing many other student-related data needs in a school. A SIS should not be confused with a learning management system or virtual learning environment, where course materials, assignments and assessment tests can be published electronically.

In this paper, a descriptive formalization of Student Management System (SMS) properties is captured using Z-notation. The choice of Z- notation as opposed to other formalization languages was based on: Z-notation decomposes specification into schema; enhancing brevity, conciseness and simplicity as opposed to other formal specification languages (Spivey, 1992; Spivey, 1998; Richard, 1998; Bart and Robert, 2001; Aneesh et al., 2003 and Jonathan, 2003). Z notation supports a large array of intrinsic and user-defined data types, as opposed to other formal specification languages (Spivey, 1992; Spivey, 1998). Z schemas describe both static and dynamic properties, as opposed to other formal specification languages (Spivey, 1992; Spivey, 1998; Richard, 1998; Bart and Robert, 2001; Aneesh et al., 2003 and Jonathan, 2003).

2 APPLIED MATERIALS

In achieving the aim of student management system formalisation; Z-notation and Unified Modelling Language (UML) serves as the core methodology tool.

Z-notation uses mathematical notation to describe in a precise way the properties a software system must possess, without unduly constraining the way in which these properties are achieved (Spivey 1998, Sannella, 1998 and Spivey, 1992). Formal specification (Mathematical notation or Z) uses mathematical data types to model data in a system and achieve it underlining objectives. These data types are not oriented towards computer representation, but they obey a rich collection of mathematical laws which make it possible to reason effectively about the way a specified system will behave. We use the notation of *predicate logic* to describe abstractly the effect of each operation of our system, again in a way that enables us to reason about their behaviour.

The other main ingredient in Z is a way of decomposing a specification into small pieces called *Schemas*. By splitting the specification into schemas, we can present it piece by piece. Each piece can be linked with a commentary which explains informally the significance of the formal mathematics. In Z, schemas are used to describe both static and dynamic aspects of a system (Spivey 1998). The static aspects includes, the state it can occupy; the invariant (quantity that is unchanged by a set of mathematical operation) relationship that are maintained as the system moves from states to state.
The dynamic aspect includes: the operation aspect that are possible, the relationship between their input and outputs; the changes of state that happen.

Software design immediately follows the requirements engineering phase (formal method) in a software process. Software design is the translation of the requirement specification into useful patterns for implementation. Unified Modeling Language (UML) is a standard modeling language used for modeling software systems. UML was utilized for the design of the student management system due to its focuse on creating simple, well documented and easy to understand software models. UML sequence diagram shows the interaction between classes (or object) in the system for each use case. The interaction represents the order of messages that are exchanged between classes to accomplish a purpose

3 APPLICATION OF FORMALISATION TO STUDENT MANAGEMENT SYSTEM (SMS)

The application of Z-notation focuses both on the static and dynamic aspect of the system. Z-notation also utilized certain basic types. The following are some of the basic types in Z [CHAR, STRING, CURRENCY, QUERY, OBJECT, COMPONENTS, BOOLEAN:: = TRUE/FALSE, DATA and OBJECT]. The student management system authentic each user using his username/ID and password on the system

The static aspects of the Student Management System (SMS) includes: Student Registration State, Student Login State, Accept Student Credentials, Present Student Result while the state variables and invariant on them includes Student: ℙ STUDENT, Registrations: ℙ REGISTRATION, Logins: ℙ LOGIN, Result: ℙ RESULT,

registered_users: STUDENT → ℙ CREDENTIALS, login_users: Student → ℙ Credentials, Present_result→ ℙ Grades. The system is initialized with no prior processes as exemplified on Figure 1

InitSmS
Sms
Sms = ∅

Figure 1: SmS Initialisation

The dynamic model for SMS defines basic operations, input-output relationship and state transitions, which includes; Student Login, Student Registration and Transition from Registration state to Login State. Figure 2 through Figure 6, shows these operations.

Add Student
Δ domRegistered_User
Student? : ℙ CREDENTIALS
Student? ∉ dom registered_users
registered_users' = registered_user ∪
{Student? ⟶ credentials?}

Figure 2: Add Student Schema

The Addstudent schema, shown on Figure 2, extends the domain of registered uses by adding newly registered students.

DeleteStudent
Δ domRegistered_User
student? : ℙ STUDENTS
Student? ∈ dom registered _users registered_users' = registered_user - ({student} credentials?)

Figure 3: DeleteStudent Schema

The DeleteStudent Schema, shown on Figure 3, shrinks the domain of registered users by removing a registered user (student) from the domain of registered users.

Student Registration
Registration : ℙ REGISTRATION
Students?: ℙ STUDENT
Credentials? : ℙ CREDENTIALS
∀k: student● credential? ∈ registration ^ registration' = registration ∪ {Student? ⟶ credentials?}
Registration = successful

Figure 4: Registration Schema

The registration schema, shown on Figure 4, captures the registration of individual students. These students usually present certain credentials identifiable by the system.

LoginStudent
Student_ID: ℙ CREDENTIALS
Student: ℙ PHYSICIAN
accepted, denied: Access ∀h: Student ● u ∈ user. access! = accepted
dom login_users ⊆ registered_users

Figure 5: LoginPhysician Schema

The login schema, shown on Figure 5, grant access to registered users, based on their access levels. The schema returns the "accepted" result for a registered user.

Student Result
grades: RESULT
result: RESULT
Ξ domResult
∃ grades: result● excellent ∈ Result ∧ result ≠ ϕ
∃ grades: result● very good∈ Result ∧ result ≠ ϕ
∃ grades: result● good ∈ Result ∧ result ≠ ϕ
∃ grades: result● fair ∈ Result ∧ result ≠ ϕ
∃ grades: result● fail ∈ Result ∧ result ≠ ϕ
∀ grades: result ● grades ∈ excellent ∧ very good ∧ good ∧ fair ∧ fail
dom sms_result⊆ result

Figure 6: Result Schema

The Student Result schema, shown on Figure 6, provide five ranking system. The predicate part receives as a request argument and returns five results utilizing received grades.

3.1 Unified Modelling Language Design of the Student Management System (SMS)

The interaction represents the order of messages that are exchanged between objects to accomplish a purpose. For the SMS system, we specify it integral part using UML sequence diagram. Figure 7 show the sequence diagram and the interaction between objects.

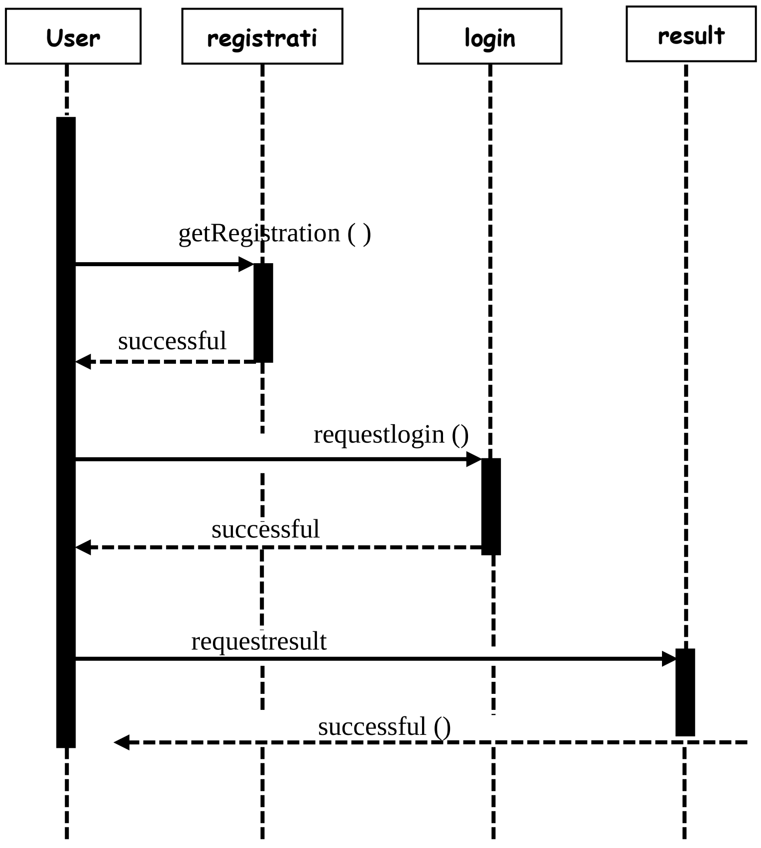

Figure 7: Sequence Diagram Modelling Student Management System (SMS)

Figure 7, models the sequence of steps involved in the Student Management System (SMS). The order of appearance of the arrows indicates the order of the activation indicating the other of flow of events / results. The sequence captures four main objects: User, registration, login and result. These objects operate an asynchronous message exchange pattern in which the response from the preceding object activates the successive objects. Therefore the response of registration must be accepted before login could be activated and the response of login must be accepted before result object could be activated.

4 FINDINGS

Based on the ease at which the users get information through the new formalised system, the following are revealed:

a. Eliminate and modify ambiguities in system properties.
b. Eliminate and ratify unknown errors
c. Support multiple system processes
d. Provide user-friendliness interface

5 CONCLUSION.

Formal specification is the bedrock in eliminating ambiguities in software systems and system requirement. In Nigeria and African as a whole this approach has not be implemented and accepted for most software system opening the avenue for system failure with huge implication such as financial loss. This research paper focuses on providing a sample representation of formal specification utilizing student management system as a case base. The system design was specified utilizing UML. The results of the finding were listed assiduously. Formal specification is an avenue for software system which must be explored in as much safety is involved.

REFERENCES

[1] Aneesh, K.., Sergiy V. and Aditya G. (2003), A Case Study of Combining I* Framework and the Z Notation, retrieved online from core.ecu.edu/vilkomirs/Papers/Vilkomir-ICEIS.pdf, April, 2015.
[2] Bart M. and Robert B. (2001), X Meets Z: Verifying Correctness In The Presence Of POSIX Threads, retrieved online from www.freedesktop.org/software/xcb/usenix-zxcb. pdf, May, 2015
[3] Bowen J. P., (1988), Formal Specification in Z as a design and documentation tool. In proc. Second IEE/BCS Conference on Software Engineering, number 290 in conference publication, pages 164-168.
[4] Bowen P., (1990), Z bibliography, Oxford University Computing Laboratory.
[5] Hunter, G. (1971), Metalogic: An Introduction to the Metatheory of Standard First-Order Logic, University of California Pres, 1971
[6] Jonathan, B. (2003), Formal Specification and Documentation using Z: A Case Study Approach retrieved online www.macs.hw.ac.uk/~gabbay/201314-F28FS/Zbook.pdf, October, 2015.
[7] Richard P. (1998), Specification and Refinement using a Heterogeneous Notation for Concurrency and Communication, retrieved online from se.inf.ethz.ch/old/teaching /ws2005 /0273/slides/formalMethods.pdf
[8] Romanovsky, A., Thomas, (2013), Industrial Deployment of System Engineering Methods. Springer-Verlag Berlin Heidelberg (2013)
[9] Rushby, J. (2000), Disappearing Formal Methods. In: High-Assurance Systems Engineering Symposium, Association for Computing Machinery (2000) 95–96

[10] Sannella D., (1988), "A Survey of formal software development methods", appeared in Software Engineering: A European Perspective, A. McGettrick and R. Thayer (eds.), IEEE Computer Society Press, pp 281-297, 1993.

[11] Spivey J. M. (1992), "The Z Notation: A Reference Manual, 2nd Edition", Prentice Hall International (UK) limited, United Kingdom.

[12] Spivey J. M. (1998), "The Z Notation: A Reference Manual", Oxford, United Kingdom.

[13] Stuart R., and Norvig P. (1995), "Artificial Intelligence: A Modern Approach", Prentice Hall (UK) International.