The International Journal of Engineering and Science (IJES)
|| Volume || 6 || Issue || 11 || Pages || PP 35-38 || 2017 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Proof of Correctness: A Formal Software System Approach

Ejiofor C. I[1]& Mgbeafuluike .I .J[2]
Chukwuemeka Odumegwu Ojukwu University [1,2]
Corresponding Author: Ejiofor C. I

----------**ABSTRACT**----------
This research paper provides an expository description of software system specification from the perspective of a diabetes diagnostic system with proof of correction in view.The papers explores Z notation as an integral formalization language in decomposing formalized diabetes system properties into associated schema. Proof of correctness has established the reliability of such system in eliminating associated errors within such system.
Keywords: *Schema, Proof of Correctness*

Date of Submission: 28-08-2017 Date of acceptance: 08-09-2017

I. BACKGROUND OF RESEARCH

Software system is concerned with the development of large, complex software system. It focuses on real-world goals for services usually constrained by precise system specification. It is also concerned with the processes, methods and tools for the development of software intensive systems in an economic and timely manner (Pamela, 2014). Software systems usually are supported by formal method built to define precisely system properties. Software system requires careful structural design.This design is usually specified using proper schema and implementation components specified using schema formation under formal methods.

Formal methods are mathematical techniques for software system specification which provide an avenue for the development and verification of software system (Gheorghe and Ancel, 2008). The uses of formal methods for software and hardware design are motivated based on the expectation in obtaining reliability and robustness in system design (Attie and Chockler, 2008). Formal methods are best described in application of broader implementation.Formal methods are mathematical techniques, often supported by tools, for developing software and hardware systems. These mathematical rigors enables user to analyze and verify model at any stage of software life-cycle such as: requirement engineering, specification, architecture, design, implementation, maintenance and testing. Formal methods also enhance data refinement involving abstraction functions and simulation proof (Attie and Chockler, 2008). Z-notation and Wright are architectural description language based on the formalization of the abstract behavior of these systems (Allen 1997). Z-notation also captures a precisely defined formal method language embedding mathematical notations (Spivey, 1997).

Software systems are error prone which often are not discovered during the development phase. A possible solution to this problem is the application of formal verification of software system through proof of correctness. The goal of the application of formal methods in program verification is to prove the correctness of software giving a mathematical proof that the software fulfills its specification. With formal proof for the correctness of a program, the needs for system programming testing are usually eliminated. Hence, the verified systems are of extreme quality as required in many industrial sectors, such as automotive engineering, security, and medical technology. However to give a formal proof one needs to have a formal specification of the software.

Therefore, it is the intent of this research paper to explore proof of correctness for diabetes diagnostic system using Z-notation.

II. APPLIED METHODOLOGY

Z-notation uses mathematical notation to describe in a precise way the properties a software system must possess, without undulyconstraining the way in which these properties are achieved (Spivey 1998, Sannella, 1998 and Spivey, 1992). Formal specification uses mathematical data types to model data in a system and achieves 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 behavior.

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 and the changes of state that happen.
The schema presented in this paper provides an avenue wherein our formal specification could be presented in fragment enabling us to associate commentaries; explaining informal the significance of the formal mathematical notation representation.

III. Z-NOTATION SCHEMA AND PROOF OF CORRECTION

An empirical expository formalization of diabetes diagnostic system using Z-notation is the focal point of this section. Z notation was explored due to the following reasons: decomposing system specification into schema enhancing simplicity and clarity (Spivey, 1992; Spivey, 1998; and Jonathan, 2003). Z notation supports a large array of intrinsic and user-defined data types (Spivey, 1992; Spivey, 1998).Z schemas describe both static and dynamic aspects of formalized system properties (Spivey, 1992; Spivey, 1998 and Aneesh et al., 2003).
Z notation comprises of certain fundamental 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 main aim of this phase is the specification of the dynamic aspect of the diabetes system with various schema, associated commentary and proof of correctness. The dynamic aspect includes physician login, physician registration and diagnosis schema. Figure 3.1 to Figure 3.5 depicts the various system schemas with associated proof of correctness.

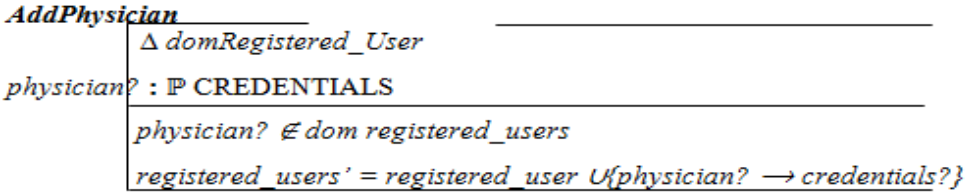

Figure 3.1: AddPhysician Schema

The *AddPhysician* Schema shown on Figure 3.1, show the extension and contrition of registered physicians.

Proof of Correctness

For correctness of formal specification, invariants must be identified existing prior and succeeding the formal statement P.

Registered_user' = dom Registered_user' *[invariant after]*
=dom (registered_user ∪{physician? ⟶ credentials?}) *[Spec. of AddPhysician]*
= dom registered_user ∪ dom {physician? ⟶ credentials?} *[fact about 'dom']*
= dom registered _user ∪{physician?} *[fact about 'dom']*
= registered_user ∪{physician?} *[invariant before]*

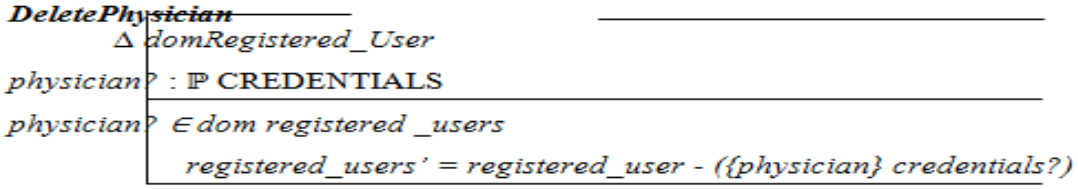

Figure 3.2: *DeletePhysician Schema*

The *DeletePhysician schema*,shown on Figure3.2, shrinks the domain of registered physician.

Proof of Correctness

For correctness of formal specification, invariants must be identified existing prior and succeeding the formal statement P.

Registered_user' = dom Registered_user' *[invariant after]*
= dom (registered_user - ({physician} credentials?) *[spec. of deletePhysician]*
= dom registered_user - dom ({physician} credentials?) [fact about 'dom']
= dom registered_user - {physician?} *[fact about 'dom'= registered_user - {physician?}*
[invariant before]

Registration

Registration: ℙ REGISTRATION

Physician?: ℙ PHYSICIAN

Credentials? : ℙ CREDENTIALS

$\forall p$: *physician* ● *credential?* $\in$ *registration* $\wedge$*registration'* = *registration* $\cup${*physician?* $\rightarrow$ *credentials?*}

Registration = successful

Figure 3.3: *Registration Schema*

The registration schema shown on Figure 3.3 provides physician registration mapping certain credentials accepted by the system.

Proof of Correctness

For correctness of formal specification, invariants must be identified existing prior and succeeding the formal statement P.

Registration = successful *[invariant after]*

= $\forall p$: *physician* ● *credential?* $\in$ *registration* *[spec. of Registration]*

= *registration* $\cup${*physician?* $\rightarrow$ *credentials?*} *[fact about registration]*

= *registration* $\cup${*physician?*} *[fact about registration]*

= *Registration* $\cup$ *(successful)* *[invariant before]*

LoginPhysician

physician_ID: ℙ CREDENTIALS

physician: ℙ PHYSICIAN

accepted, denied: *Access*

$\forall u$: *physician* ● *u* $\in$ *user. access!* = *accepted*

dom login_users $\subseteq$ *registered_users*

Figure 3.4: *LoginPhysician Schema*

The login schema, shown on Figure 3.4, grant access to registered users, based on their access levels. The schema returns the "accepted" result for a registered user.

Proof of Correctness

For correctness of formal specification, invariants must be identified existing prior and succeeding the formal statement P.

Login_user' $\subseteq$ *dom register user'* *[invariant after]*

=*dom (login_user* $\subseteq${*physician?* $\rightarrow$ *credentials?*}) *[Spec. of LoginPhysician]*

= *dom login_user* $\subseteq$*dom {physician?* $\rightarrow$ *user.access}* *[fact about login]*

= *dom login_user* $\subseteq${*physician?*} *[fact about login]*

= *login_user* $\subseteq${*physician?*} *[invariant before]*

PhysicianDiagnosis

symptoms: DIAGNOSIS

diagnosis: DIAGNOSIS

Ξ *domDiagnosis*

$\exists$*symptoms: diagnosis* ● *Type I Diabetes* $\in$*Diagnosis* $\wedge$ *diagnosis* $\neq \phi$

$\exists$*Symptoms: diagnosis* ● *Type II Diabetes* $\in$*diagnosis* $\wedge$*diagnosis* $\neq \phi$

$\forall$*Symptoms: diagnosis* ● *symptoms* $\in$*Type I Diabetes* $\wedge$*Type II Diabetes* $\in$*diagnosis*

dom hscm_pt_diagnosis $\subseteq$ *Diagnosis*

Figure 3.5: *Diagnosis Schema*

The physician Diagnosis shown in Figure 3.5 provides two class of diagnosis based on presented symptoms. The predicate part receives as a request argument and returns two diagnostic results utilizing received symptoms.

Proof of Correctness

For correctness of formal specification, invariants must be identified existing prior and succeeding the formal statement P.

dom hscmpt_diagnosis' ⊆ Diagnosis'	*[invariant after]*
= dom (hscm_pt_diagnosis ⊆{diagnosis? ⟶ symptoms})	*[spec. of diagnosis]*
= dom hscm_pt _diagnosis ⊆{diagnosis? ⟶ diagnosis})	*[fact about diagnosis]*
= dom hscm_pt_diagnosis ⊆{diagnosis? ⟶Type I diabetes})	*[fact about diagnosis]*
= dom hscm_pt_dignosis ⊆{diagnosis? ⟶Type II diabetes })	*[fact about diagnosis]*
= dom hscm_pt ⊆{diagnosis?}	*[fact about diagnosis]*
= hscm_pt_diagnosis ⊆{diagnosis?}	

IV. DISCUSSION

The application of formal proof or proof of correction has been devoid from numerous software system specifications enhancing system with system errors. The diabetes diagnostic system has been formalized with associated schemas, commentaries and proof of correction. The formalization specifing the properties of the diabetes diagnostic system such as: add physician; delete physician, login, registration and diagnosis captured through various schemas. The schemas were also complemented using associated commentary describing the underlining purpose of the system. Proof of correction attached to each schemas has help established the reliability and accurate of these formalized modules.

V. CONCLUSION

The quest for a formal approach with proof of correctness in establishing the correctness and reliability of software system has been exemplified within this research paper using diabetes asan empirical case study. This research paper captured five schemas with associated commentary and proof of correctness establishing the correctness of formal specified properties. It is hoped that with this expository paper, formal proof will be enshrined in future formalized approaches.

REFERENCES

[1] Allen, R. (1997), A formal approach to software architecture. Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Issued as CMU Technical Report CMU-CS-97-144.

[2] 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.

[3] Attie P. C., Chockler H., (2008) "Automatic verification of fault-tolerant register emulations", Electronic Notes in Theoretical Computer Science, vol. 149, no. 1, pp. 49–60.

[4] 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.

[5] Pamela Z. (2014), Software System Engineering, retrieved online from http://www0.cs.ucl.ac.uk/staff/A.Finkelstein/defn.html.

[6] Gheorghe, A. V., & Ancel, E. (2008), Unmanned aerial systems integration to National Airspace System. In Infrastructure Systems and Services: Building Networks for a Brighter Future (INFRA), 2008 First International Conference on (pp. 1-5), IEEE.

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

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

Ejiofor C. I Proof of Correctness: A Formal Software System Approach." The International Journal of Engineering and Science (IJES), vol. 6, no. 11, 2017, pp. 35-38.