Proc. of the Eighth Intl. Conf. on Advances in Computing, Communication and Information Technology - CCIT 2019

ISBN: 978-1-63248-169-6 DOI : 10.15224/978-1-63248-169-6-12

Adoption of Formal Methods in the Commercial World

[Aifheli Nemathaga and John Andrew van der Poll]

***Abstract* — There have been numerous studies on formal methods but yet there is diminutive utilization of formal methods in the commercial world. This can be attributed to many factors such as that few specialists know how to use Formal Methods (FMs) and also the use of mathematical notation gives a perception that formal methods are hard. FMs have been used in the software development world since 1940 during the earliest stage of computer development. To date there is a slow adoption of FMs and they are used mostly in mission critical projects such as in the military and aviation. In this paper we focus on how to increase the pace of FMs adoption in the commercial world. As part of this work a framework is established to facilitate the use of FMs in the commercial world or commercial systems. A mini ERP system specification is presented in both an informal technique and a formal notation to demonstrate how a formal specification can be derived from informal specification guided by the Enhanced Established Strategy for formal specification.**

***Keywords* — Commercial system, ERP, First-order logic, Formal Methods (FMs), Formal Specification, Formal Verification, Set theory, TLA+, Z, Zermelo-Fraenkel**

I. Introduction

This paper investigates the feasibility of utilising Formal Methods (FMs) within commercial software development. In addition to the findings it defines and develops a framework to facilitate the use of FMs in commercial software development. The paper places the focus on ERP systems and a small formal methods specification is written specifying requirements for an ERP system.

This paper is organized as follows: Section II provides an overview and context of formal methods. Section III of the paper is about formal methods adoption further explaining A – the reason for slow adoption and B, the differences between FMs and natural language/prose. Section IV is about formal methods in practise which leads to Section V which in turn gives practical examples of formal methods in the commercial world. Section VI presents a formal methods adoption framework and Section VII concludes the paper and gives directions for future work in this area.

Aifheli Nemathaga
School of Computing (SoC), Florida Campus
University of South Africa
South Africa

John Andrew van der Poll
Graduate School of Business Leadership (SBL), Midrand Campus
University of South Africa
South Africa

II. FMs Overview and Context

The advancement of hardware during the past 30 years has led to the development of large and complex systems. The growing technologies range from mobile devices, industrial machinery and automobiles. These systems require fast processing in order for hardware and software to work together to perform complex tasks [44]. The lines of code have increased from a couple of lines to 40 million lines in software and it is still increasing. As these systems grow designers and engineers face many challenges. These systems are designed, enhanced and modified often during their lifetime. Software development is time consuming and a costly process, and research has shown most software do not meet users' needs and is delivered out of their respective budgets [6]. This also applies to ERP systems, that is ERP project implementations are mostly unsuccessful or implemented out of timelines and with higher costs [39]. Consequently, many software development techniques have been developed to try to overcome these challenges.

Formal methods have shown to be one of the promising techniques to potentially overcome some of the above challenges. There are numerous benefits in using formal methods, e.g. they have been shown to reduce the number of defects in software development [1]. In the software development world there is always a search to find better ways of developing software that are free from errors and delivered within timelines and on budget. This led to the development of various frameworks and methodologies of software development. The most famous and widely used is the traditional waterfall methodology which proposes that software has to be developed using a stepwise approach, i.e. requirements, design, implementation, verification and maintenance [31]. Each stage must be finalised prior to starting the next. Waterfall is one of the oldest models still used today [28]. Yet, many of the waterfall projects are delivered out of budget, with many defects and the end product usually does not present the real needs of the user [29]. There is an increased uptake of the Agile methodology in the commercial world; software is developed in increments and in rapid cycles. Agile's main objective is to deliver value to customer by means of working software [4]. An agile methodology is guided by a manifesto which defines the principles that must be followed when using Agile. That said, Agile has many disadvantages such as a lack of documentation and the project can easily be taken out of track if a customer's requirements are not well understood. Given the aforementioned, formal methods can plausibly be incorporated during any stage or phase of the SDLC and has proven to reduce the error count [14].

III. Formal Methods Adoption

Software testing has traditionally been the only technique that has been used and is still used to find defects. Yet, code testing is not an effective way of finding subtle

Proc. of the Eighth Intl. Conf. on Advances in Computing, Communication and Information Technology - CCIT 2019

ISBN: 978-1-63248-169-6 DOI : 10.15224/978-1-63248-169-6-12

bugs/error in the design. The use of formal methods helps to reduce errors early in software development, thereby saving on the cost of software projects. Formal methods are categorised mainly in two groups, namely, (1) Pure mathematics which is challenging and is mostly not used in the real world, and (2) Software engineering which focuses on creating high quality software [42].

Formal methods use mathematics to analyze and verify models at any stage of the software development process [45]. One of the most significant parts of the development process is to understand the needs of the users. According to George and Vaughn [14] formal methods are useful when gathering, articulating, and representing requirements. This then assists the developer in developing a system that meets a user's needs.

Another type of formal methods is the state-based method, which involves the creation of state machine specifications, simulation proofs and abstract functions. During development formal methods are used to verify code by attempting to prove theorems (discharging proof obligations) about the proposed system. Some tools used for formal methods can automatically generate compilable code e.g. the B-method. The clarity, completeness and consistency of a formal specification facilitate the derivation of test cases [41]. As part of this paper a formal specification will be documented using the Z notation which is a formal language as indicated in Figure 1.

	Sequential	Concurrent
Algebric	Larch (Guttag, et al., 1993) OBJ (Futatsugi, el al., 1985)	Lotos (Bolognesi and Brinksma, 1987),
Model-based	Z (Spivey, 1992) VDM (Jones, 1980) B (Wordsworth, 1996)	CSP (Hoere, 1985) Petri Nets (Peterson, 1981)

Figure 1. Formal Specification Languages [34].

A. *Reasons for Slow Adoption*

Many software development institutions don't consider it to be cost-effective to incorporate FMs in their software development processes [34]. Some of the stumbling blocks in the use of FMs in the commercial world is the perception that requirements formalization is difficult and the creation of a formal specification as part of the formal methods process is error prone and time consuming [3]. Formal methods are based on mathematical notations, which are perceived as being hard, but in reality the notation can readily be mastered and used. For example, it is easier to learn such notation than learning a new programming language [7].

Another reason for the slow adoption of FMs is that most engineers also view FMs to be a mechanism that is practically hard to understand and utilise [35]. In the same vein, the commercial world or businesses are of the view that using FMs will increase the cost of system development due the level of training required. Education plays a major role to individuals developing and designing systems. In addition also management need to be educated if they are to successfully apply formal methods within their organizations [45].

The above ideas lead to the following proposition to be used in the construction of our formal-methods adoption framework:

Proposition (Prop) 1: *Education plays a major role in formal methods adoption. This includes educating from the high school level to the university level as well as organisational training in formal methods. Such education plays a pivotal role in the adoption framework.*

As more software development processes gain popularity e.g. the Agile methodology, there is a view that formal methods do not support other software development processes. Yet, formal methods can be beneficial in every step of the software development life cycle as they assist in alleviating unclear and unrealistic requirements at the start of the development process, leading to the production of a high quality product with fewer defects [11].

The lack of easy step-by-step guidelines on how to use formal methods also contributes to the slow adoption. Many developers view formal methods as being limited to academic projects for tertiary education. Bowen & Hinchey [8] postulated that tools, standards, and education would make or break commercial adoption, while some observed differences among academics who view FMs as "inevitable".

Most traditional software development techniques are established and proper standards have been set. Tools supporting those techniques are widely accepted and used in business [35]. On the other hand, FMs appear to have inadequate tool support. Some formal methods tools do not work suitably with development/programming tools. Formal methods tools are also not seen as facilitating the user experience (UX) [44].

When compared to traditional techniques there are many certifications that one can acquire and many institutions offer training around such techniques. The study done by Davis [10] shows that formal methods adoption may also be attributed to certification authorities not having enough education on how to appraise FMs artefacts and they are not highly informed of formal methods benefits and underlying techniques.

The above discussion leads to further insight into Proposition 1 as follows:

Prop 1.1: *In addition to the above proposition formal certificates and diplomas in formal methods should be created and awarded to those who qualify. Certification authorities should be well informed about the benefits of formal methods*.

Usually when developing a system for clients, users review and sign off the requirements specification i.e. the Business Requirement specification (BRS) or Functional Requirement Specification (FRS). The reviews are for validation and ensuring all user requirements are included in the specification [18]. The specification can then be used to bill an external client; for an internal client an agreement could confirm that the stated requirements will be developed. Clients find it hard to review formal specifications due to the mathematical notations used, resulting in project delays.

Proc. of the Eighth Intl. Conf. on Advances in Computing, Communication and Information Technology - CCIT 2019

ISBN: 978-1-63248-169-6 DOI : 10.15224/978-1-63248-169-6-12

There is also a psychological and human resource factor with the slow adoption of FMs in business. Within the organization or business some people just do not like formalism; the same applies to formal methods as some engineers, especially those who are already in an agile environment, will be more reluctant to use FMs [35]. In the commercial world the development of some projects are relatively fast, so there is little time to conduct a proper formal analysis. In today's world individuals change positions frequently, for example from a software engineer to a manager or even changing companies. This results in having to upskill a new employee which is time consuming.

Consequently we arrive at our next proposition:

Prop 2: *Buy-in from all the business stakeholders is necessary for FMs adoption. Getting Top level management to agree and accept the use of formal methods may well result in the whole organisation adopting formal methods.*

As indicated, there are numerous misunderstandings with formal methods, leading to slow adoption in business. Businesses view formal methods as a technique that places too much emphasis on the theory, rather than real world applications. Another huge misconception is that if FMs are used, then there is no need for testing; this relates to one of the 7 FMs myths – the use of a formal method guarantees that the resulting software or system is perfect [20].

There is a huge gap between the real world and formalism, namely, transforming clients' requirements from informal requirements to formal requirements requires serious clarification of the problem [24]. Generally, there is no widely accepted principle or guidance of eliciting a client's requirements and how to specify them using a formal specification language [20].

Sommerville [34] indicated four (4) reasons of why there is a slow adoption of FMs in the commercial world:

1. The utilization of other system engineering techniques, e.g. configuration management and structured techniques have improved software quality.
2. Lately, software is developed and delivered fast. The main focus is time to market (TTM) rather than quality; in some instances customers will accept software with some errors so long as it can be delivered rapidly. Rapid software delivery does not work well with formal methods.
3. The narrow scope of formal methods often does not cater for user interface design and user interaction.
4. Developing a formal specification for a system upgrade becomes a time consuming and costly process, aspects which the commercial world are unlikely to compromise on.

The above leads to the following proposition:

Prop 3: *Widely accepted principles and guidelines on FMs can improve the adoption thereof. Practical, real world examples of FMs successes and failures must be published in the software engineering and management communities.*

B. *Differences between formal and informal (natural language) specifications*

TABLE I. DIFFERENCES BETWEEN FORMAL AND INFORMAL SPECIFICATIONS [19].

Informal (natural language)	Formal Methods
Each stakeholder has his/her own interpretation of the requirements.	There is generally a complete and broad view of system requirements.
More errors present, and if not corrected can result in high project costs.	Fewer errors and omissions in the specification document.
Uses a combination of graphics and semiformal notations.	Uses mathematical notation, first-order logic and natural language prose.
Little or no use of Mathematics. General knowledge on the software engineering domain used.	Specifiers and stakeholders ought to be familiar with the mathematical notation.
Specifiers leave room for inconsistency and ambiguity.	Provides conciseness, clarity and unambiguity.
They are ideal for eliciting requirements.	Allows the software engineer to produce high quality systems.

Both formal notations and informal techniques can result in a vague understanding of the system [24]. All this depends on the software engineer or the developer understanding what to build irrespective of the language used in the specification. Depending on the specifier's willingness, it is possible to learn formal languages but it takes time and may be a costly exercise.

IV. FMs in practice

For this paper we will show how FMs are written and the chosen language for this paper is Z [36]. The commercial system that will be specified formally is an ERP system.

A brief discussion of what an ERP system is follows next.

A. *What is Enterprise Resource Planning (ERP)?*

As per Seo [32] "ERP is a software architecture that is designed in order to expedite the information flow as well as the information sharing between various departments in a company, and also to provide to decision-makers an enterprise-wide view of all information that they may need to assist them in decision-making".

From the above description it follows that Enterprise Resource Planning (ERP) systems are combined software programmes that are clustered into standard functional modules i.e. Procurement, Human resources, Finance, Contract management, Customer relationship management (CRM), etc. developed by a Vendor or in-house [26]. Usually a single database, or more generally a data warehouse, together with a unified interface across the entire enterprise is utilized [2]. Some ERP software can be purchased off the shelf and customized afterwards to meet specific customer needs. An ERP system assists business in performing their daily operations which can bring about

Proc. of the Eighth Intl. Conf. on Advances in Computing, Communication and Information Technology - CCIT 2019

ISBN: 978-1-63248-169-6 DOI : 10.15224/978-1-63248-169-6-12

substantial benefits within the organization. Despite all the benefits mentioned above, ERP project implementations are mostly unsuccessful or implemented out of timelines and with higher costs than budgeted [39].

Failure of ERP project implementation can be attributed to different factors such as unclear requirements, project managers focusing on the financial aspect of the business and neglecting other parts of the project and no proper software development process in place to manage the projects to name a few. Most of the time the success of the project is attributed to delivering the project within the timelines and on budget; developers and managers tend to forget about the users of the system and the smooth change from a previous (existing) process to the new one [16]. Most of the aforementioned failures do apply when implementing other software such as CRM, Billing systems etc. The researchers believe the use of formal methods will help alleviate many of the problems during ERP implementation in an organization.

1) ERP Modules

The ERP architecture in Figure 2 presents some of the main modules that are contained in an ERP system.

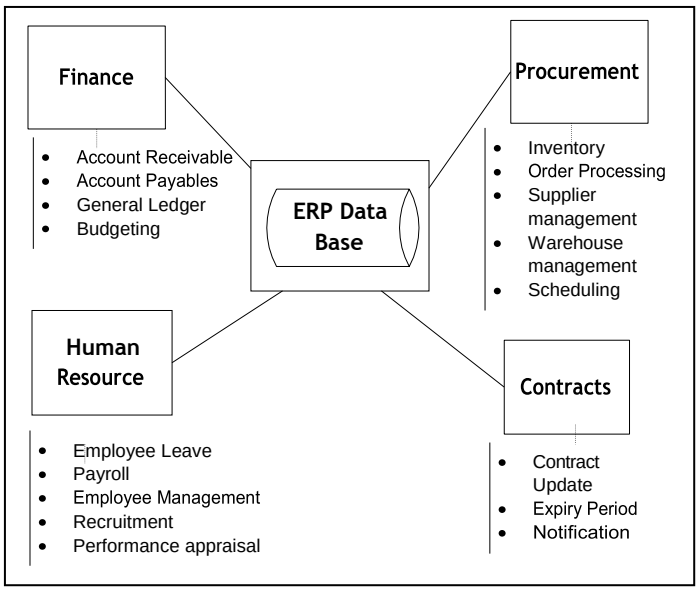

Figure 2. An ERP Architecture [22] .

The focus of this paper will be on the *procurement* module otherwise known as the purchasing module. The reason for this is that a procurement module is usually the most widely used module in an organisation [13].

The Procurement module deals with the purchasing of materials for internal use or resale within the organization. Procurement mostly have a built-in workflow which can automatically evaluate the supplier, measure the inventory at hand which is otherwise known as warehouse management. Lastly, most of the purchasing modules are integrated into invoice verification. Gao [13] calls the procurement module the "internet procurement".

B. *The Z Specification Language*

The Z specification language was established in the late 1970s at Oxford University by the Programming Research Group otherwise known as the PRG [11]. Z is constructed on a strongly typed fragment of Zermelo-Frankel (mathematical) set-theory and first-order logic; it embodies a rich notation. Using a formal specification language such as Z, software systems can be designed with little uncertainties [17]. Many formatting tools such as Latex and type-checkers for writing syntactically correct Z specifications have been developed since Z is written mostly in non-ASCII mathematical symbols (refer to the Z schemas that follow).

A Z specification comprises of schemas with narrative text in-between. A schema is an organizing unit that contains logically associated mathematical notation.

Below is the generic format of a Z schema:

SchemaName

Declarations

$Predicate_1;\ldots;Predicate_n$

Apart from a few exceptions, a Z schema is usually divided into 2 parts as follows. Part one above the short (middle) dividing line specifies the variables (components) and their types (declarations) to be used in the specification. Part two specifies, in first-order notation, the constraints on, and relationships among the components of the specification.

1) Some of the tools that are used for a Z specification

Naturally, tool support assists a lot when developing a Z specification. One of the add-on advantages of Z over some other languages is a number of software tools that have been developed to assist with the construction of a specification [42].

- CadiZ – Computer aided design in Z by Mcdermid and Toyn [40], used to construct, type check and reason about a Z specification.
- Fuzz – Mike Spivey's commercial type checker for Z [36]. More information is available at the URL: http://spivey.oriel.ox.ac.uk/corner/Fuzz_typechecker_for_Z.
- The Community Z Tool (CZT) by Utting and Malik [25] (http://czt.sourceforge.net/) is another useful addition to the arsenal for writing Z specifications.

Using a formal notation assists in understanding how the system will operate and it allows the developer to have more choices about the design of the system [17]. The omitted parts of the specification become easy to identify and the overall quality of the specification document is enhanced.

The above discussions leads to our next proposition:

Prop4: *Tools that are readily available and up to date with the lasted technology should facilitate the adoption of FMs. Such tools should be integrated with the requirements management software and standard software programming tools e.g., MS Visual Studio.*

The development of the Z specification presented below will be guided by the Enhanced Established Strategy (E^2S)

Proc. of the Eighth Intl. Conf. on Advances in Computing, Communication and Information Technology - CCIT 2019

ISBN: 978-1-63248-169-6 DOI : 10.15224/978-1-63248-169-6-12

created by Van der Poll and Kotze [43]. The Enhanced Established Strategy stems from the Established Strategy for constructing a Z specification and gives guidance and principles to the software engineer when writing a Z specification.

2) Procurement requirements

Partial requirements of our ERP system in this paper are depicted in TABLE II.

TABLE II. PROCUREMENT MODULE REQUIREMENTS LIST

No	Formal Methods
1	The system must be able to keep track of stock for several products
2	Product should have a name, price and quantity of available stock recorded on the system.
3	Each product must have a unique name.
4	User should be able to update products name, price and quantity of stock on hand.
5	The system should have the ability to produce a report with all the products that are below set threshold.
6	The system should allow for the capturing of orders.
7	Once a new order for a specific product is captured it will stay on the "pending" status.
8	All orders on the pending status can be deleted, once deleted the status should change to "Cancelled".
9	A quantity of order should always be more than one.
10	A record of the quantity, price and product name order must be kept.
11	All orders with the status that is pending should be processed when there is required stock on hand.
12	Once the order is processed the status should change to "processed" and the quantity should decrease with same number if products ordered.
13	Customer information needs to be stored and linked to the order. Information includes the name, address and phone number must be stored.
14	One customer can have multiple orders

3) Formal Specification of the Purchasing Module

State Space of Product Schema

Schema *Product* below specifies aspects of products in the ERP system (recall emphasis will be on the procurement part of the ERP).

Product

$products : \mathbb{P}\ PRODUCT$

$produName : PRODUCT \rightarrowtail STRING$

$produPrice : PRODUCT \rightarrow AMOUNT$

$proQuantity : PRODUCT \rightarrow \mathbb{N}$

$\text{dom}\ produName = products$

$\text{dom}\ produPrice = products$

$\text{dom}\ proQuantity = products$

Product Schema summary

$\mathbb{P}\ PRODUCT$ represents characteristics of all product detail in the system. Further specification of these characteristics is beyond the scope of this paper.

prodPrice: $PRODUCT \rightarrow AMOUNT$ is an attribute (monetary amount) of the product and it is declared as a partial function notation ($\rightarrow$).

prodName: $PRODUCT \rightarrowtail STRING$ denotes the name of the product. Requirement no 3 in TABLE II states that no two products can have the same name, hence a partial injective function is used to specify product names.

The domains are specified in the predicate part of the schema. In this requirement each attribute of the product should equal the identities collection. i.e.

$dom\ produName = products$

$dom\ produPrice = products$

$dom\ proQuantity = products$

State Space of Customer Schema

Before creating an Order schema a customer schema ought to be specified. A customer is linked to an order.

Customer

$customers : \mathbb{P}\ CUSTOMER$

$custoAddress : CUSTOMER \rightarrowtail STRING$

$custoPhone : CUSTOMER \rightarrowtail STRING$

$\text{dom}\ custoAddress = customers$

$\text{dom}\ custoPhone = customers$

Customer Schema summary

Definition *customers*: $\mathbb{P}\ CUSTOMER$ represents the set of all existing customers in the system.

Definitions *custoAddress:* $CUSTOMER \rightarrowtail STRING$ and *custoPhone:* $CUSTOMER \rightarrowtail STRING$ are attributes of customers and are specified using a partial function (denoted by $\rightarrowtail$). For reasons of space the Order schema which links an order to a customer is not shown in this paper.

The above formal specification fragment shows how the procurement module can be specified using a formal method, in this case Z. Some other formal notations are VDM, B, CSP, OBJ, etc.

Aspects of formal specification illustrated in this section lead to proposition 5:

Prop 5: *Using the Z notation as an entry language ought to facilitate the adoption of formal methods. Z is believed to be easy to learn and apply. Only basic mathematical set theory and logic are required.*

Proc. of the Eighth Intl. Conf. on Advances in Computing, Communication and Information Technology - CCIT 2019

ISBN: 978-1-63248-169-6 DOI : 10.15224/978-1-63248-169-6-12

V. Formal Methods in the commercial world

Since the development of formal methods in the 1980s, their adoption or use within the business arena is slow [10]. Yet, the following software and hardware giants are known to be using Formal Methods:

- Amazon
- Intel
- NASA
- NATS
- Xilinx

Other companies known to also use FMs are: Qualcomm, Nvidia, Cisco, Broadcom, Samsung, Mediatek, AMD, and Huawei. Originally Google's and Microsoft's main foci were software but they starting to develop their own hardware and they have since also adopted formal methods [9]. Start-ups are slowly picking up formal methods as these provide a good return on investment (ROI) with clean code, hence less money is spent on rectifying defects [30].

Next we elaborate on the successful use of FMs by the Intel company

Intel's core business is hardware; for hardware to function correctly, the following needs to be developed: Microcode, Firmware, Protocols and Software. In almost all the products Intel provide, they used to experience problems with the diversity of verification [12]. Consequently, Intel developed various solutions in an attempt to solve verification problems. Their solutions include Propositional Equivalence Inspection (PEI), Symbolic simulation, Symbolic Trajectory Evaluation (STE) and temporal logic model checking.

Intel experienced numerous challenges with some of their products, the most challenging was a physical problem which was the overheating of their Chips and the FDIV bug, which could readily be solved through the use of formal methods. Intel invested over $147 million to cover the cost incurred from chip overheating and the verification problems which also led to the improvements of formal methods within Intel. Intel has realised numerous benefits with using formal methods and they continue to use them on many projects [9].

The above discussions lead to our next proposition:

Prop 6: *Publications of formal methods successes in terms of the money saved in projects, clear specification produced and the overall final product delivered with fewer defects will raise much interest needed for the adoption of FMs.*

Next we combine our propositions and above observations into a formal-methods adoption framework.

VI. Formal Methods Adoption Framework

The proposed framework will be based on the propositions made in this paper and the work that has been done in this knowledge area by other researchers. We first discuss the framework in a tabular format then a graphical presentation of the framework is presented.

A. *Adoption Framework Table*

A framework can be defined as a skeleton or a basic structure of the underlying system [5]. This can be updated as required by adding or deleting items. From a software perspective it can be defined as a set of functions within a system and how they interconnect.

TABLE III presents the proposed framework on the strength of the propositions identified throughout the paper and this assisted in creating our Formal Methods Adoption Framework.

TABLE III. ADOPTION FRAMEWORK

ASPECT	DISCUSSION
Education	Software engineering education in the early stage
	Introduction to formal methods for first year university students
	Universal formal methods standards
	University accreditation specifically on formal methods
	Set theory basics at an early stage of educationssystems
	Step by Step guide on transforming informal requirement to formal specification
	Knowledge sharing and common terminology
Buy-in	Public sector using formal methods for their systems
	Enterprise Top management buy-in
	Project Manager and Senior managers buy-in
	Training companies
	IT community buy-in
	Formal Method language e.g. Z
Remuneration	FM specialist salaries, Scare skill
Environment	IT environments where FMs are going to be utalised
	Tools to write formal specification
	Integration of MS office to formal specification languages
	Open source tools
	Collaborative environment for formal methods specialist to meet
	Built the right attitude within teams, Team buildings
Support Tools	LaTeX, Fuzz,
Publications	Successful use of formal methods should be published regularly
	Forums i.e. internet news letters of formal methods
	Encourage use of formal methods on open source systems
	Library catalogue on formal methods
Results	Positive and negative results should be made available
	Description of each successful component of the system built using formal methods
	System developed using formal methods used in real business environment
	Positive and negative results should be made available

Proc. of the Eighth Intl. Conf. on Advances in Computing, Communication and Information Technology - CCIT 2019

ISBN: 978-1-63248-169-6 DOI : 10.15224/978-1-63248-169-6-12

B. *Adoption Framework Diagram*

Figure 3 is the graphical presentation of the framework in TABLE III. All the steps can be performed in parallel. The larger box represents that the more we focus on that step the higher the probability that the adoption will be a success.

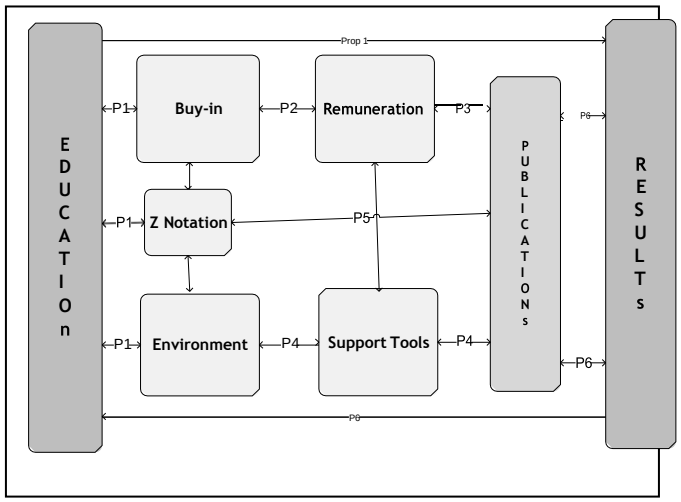

Figure 3. Formal Merhods Adoption Framework

The contents of Figure 3 are discussed next.

Education: This all important aspect forms the foundation to allow for the adoption of FMs in the commercial world. Without dedicated education and the transfer of skills to the new and upcoming engineers the current state of formal methods may well stay as–is. As observed in TABLE III above, education should be from a level as early on as high school. A basic introduction to formal methods should be given, followed at university level by a module on formal methods for first year students. Education will also go a long way to solve the problem of not having a common terminology in formal methods. Students who qualify or pass this course should be recognised by awarding an appropriate accreditation. The Z notation is relatively easy to teach and learn as it requires basic mathematical knowledge. Students and IT professionals might start off by first learning Z and then be introduced to other formal specification languages over time.

Buy-in: IT professionals should be encouraged to use and invest in the use of FMs within their organisations. Buy-in is needed from top management right through to the project manager and the software engineers. Public enterprise buy-in to utilize formal methods in their systems is also important. Furthermore, buy-in from companies that provide IT training courses and the IT community in general will have a major impact on the promotion of formal methods. Established professional societies (e.g. the IEEE) can provide standardized FM teachings in the use and practical application of formal methods. To secure the necessary buy-in from software engineers, training or awareness of formal methods should be provided in a top down approach; i.e. from top management to senior managers, then analysts and then developers. With Z as a recommended formal-methods language for this research, top management need to understand the benefits that Z brings, such as it is easy to comprehend and also the flexibility it inhibits to model a specification that can lead directly to code through successive transformations, and so forth. Many specifiers are familiar with the Z syntax and semantics, hence it's arguably the most used formal language [7].

Remuneration: Good compensation to FM specialists will also attract more people to join the formal methods arena. It should also motivate students and professionals already in the computer science industry to obtain certification in formal methods. Most companies nowadays are concerned with cutting cost and that's what top management understands to place as a priority. Hard evidence of cost saving, reduction in development time and the improved quality of the resultant system when using FMs, should be made available to portfolio- and programme managers.

Environment: Equally important is the environment in which formal methods are to used. The environment should encompasses proper tools to facilitate the use of formal methods. Integration of current software engineering tools with formal methods tools to allow for a smooth transition should be put in place. Teams working on formal methods must have the right attitudes which can also be influenced by the environment they are working in. An encouragement of the use of formal methods in Open source software will bring about new ideas and engender a positive attitude and perception towards formal methods.

Support tools: FMs tools should enhance the UX (i.e. be user friendly) and should allow for an easy integration into current development environment such as the .NET framework and Linux development frameworks. It should also be possible to integrate FMs artefacts and techniques into an existing SDLC. As more and more computer devices are mobile e.g. smart phones and tablets, more Apps for these devices should utilise some form of formalisation. FMs tools should have more automation as well as automated test generation. Lastly, formal methods should facilitate clear estimations on how long it will take to perform analyses. For example, Van der Poll [42] states that "it must be possible to continuously measure the progress archived by formal technique during software development".

Publication: Regular publications on formal methods usage should be encouraged. This can be made available via internet news letters, forums, blogs and a public library (Library catalogue on formal methods). As we can see from the the FMs adoption framework in Figure 3, buy-in and environments feeds into publication(s). A positive buy-in and conducive environment should lead to the successful implementation of formal methods – these should be publicised. In South Africa we have an institute called the CSIR (Council for Scientific and Industrial Research) which is a leading technology research organisation in the country. This body can be used to promote the use of formal methods in software development.

Results: after all the steps have been followed, positive results and findings should be the output. Even negative results should be made known and lessons should be learned through these. Results should lead to the development of the systems using formal methods in the practical world. Each successful component should be described and how success was archived.

Proc. of the Eighth Intl. Conf. on Advances in Computing, Communication and Information Technology - CCIT 2019

ISBN: 978-1-63248-169-6 DOI : 10.15224/978-1-63248-169-6-12

VII. Conclusions and Future work

The use of Formal Methods for software development has been shown to be beneficial, yet their adoption in commercial software development remains slow. Formal methods are still viewed as being difficult owing to their use of mathematical notations. Also, there are many myths that surround the use of formal methods, such as they guarantee a perfect system. Putting more emphasis on education and results will help in the increased uptake of formal methods commercially.

Tools that are readily available and enhance the user experience (UX) will not only encourage specifiers and software engineers to embark on formal methods but aid in producing specifications and systems that are most likely error free. Positive results on the successful utilization of FMs in the commercial system should be made publicly available.

More research and development need to be undertaken to promote the commercial uptake of FMs. Both the public and private sectors should be encouraged to engage with these processes.

A. *Future work*

Formal methods have been around for years and this paper focused on what could be done to increase formal methods usage in commercial world. While this paper made a contribution towards the uptake of FMs through an adoption framework, much work remains to be done.

Common terminology ought to be developed across the academic and industrial formal methods fields. This needs to be widely accepted and standardized across the board. The issue of formal methods tools have been cited by many researchers, tools needs to be developed and these include automated and semi-automated reasoners to discharge proof obligations (POs) resulting from formal specification constructs. Existing tools are either not user friendly or do not perform all the required functionality to write formal specifications [21]. Such tools need to be integrated with current development frameworks such as .NET and JAVA to name a few. Automatic conversion of first-order logic specifications to a full Z specification needs to be looked at. Tools need to be classified and demonstrations of the tools in the form of videos (vodcasts), also indicating the strength of it in practice, ought to be made.

Immediate future work following this paper will be in the form of a survey among academics and practitioners to validate and enhance the framework in Figure 3. Such instrument to assist business in choosing the correct off-the-shelf FMs tools must be made available to business to assist them in their IT operations. More practical examples are needed specifying the advantages and disadvantages of different FMs design methodologies.

Amongst others, the following questions need to addressed:

- How can a formal specification be validated with the user and other stakeholders?
- How can a formal notation be sensibly integrated with more widely used notations such as UML, its class diagrams and use case diagrams?
- How can one automate formal descriptions to generate test cases or even code?

Without proper education and transfer of skill to the new upcoming software engineers the current state of formal methods will stay as–is.

References

[1] Adesina-Ojo, A. (2011). Towards the formalisation of object-oriented methodologies. University of South Africa, Pretoria. Retrieved from http://hdl.handle.net/10500/11957

[2] Al-Ghofaili, A. A., & Al-Mashari, M. A. (2014). ERP system adoption traditional ERP systems vs. cloud-based ERP systems. In 4th International Conference on Innovative Computing Technology, INTECH 2014 and 3rd International Conference on Future Generation Communication Technologies, FGCT 2014 (pp. 135–139). https://doi.org/10.1109/INTECH.2014.6927770

[3] Atlee, J. M., Beidu, S., Day, N. A., Faghih, F., & Shaker, P. (2013). Recommendations for improving the usability of formal methods for product lines. In 2013 1st FME Workshop on Formal Methods in Software Engineering, FormaliSE 2013 - Proceedings (pp. 43–49). https://doi.org/10.1109/FormaliSE.2013.6612276

[4] Beck, K., Beedle, M., Bennekum, A. Van, Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). Manifesto for Agile Software Development. Retrieved from http://agilemanifesto.org/

[5] Börger Egon, & Stärk Robert. (2003). Abstract State Machines. Springer-Verlag Berlin Heidelberg, 437. https://doi.org/10.1007/978-3-642-18216-7

[6] Bourque, P., & Fairley, R. E. (2014). SWEBOK v.3 - Guide to the Software Engineering - Body of Knowledge. IEEE Computer Society. https://doi.org/10.1234/12345678

[7] Bowen, J. P. (2016). The Z notation: Whence the cause and whither the course? In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 9506, pp. 103–151). https://doi.org/10.1007/978-3-319-29628-9_3

[8] Bowen, J. P., & Hinchey, M. (2012). Ten commandments of formal methods… Ten years on. In Conquering Complexity (pp. 237–251). https://doi.org/10.1007/978-1-4471-2297-5_11

[9] Cousineau, D., Doligez, D., Lamport, L., Merz, S., Ricketts, D., & Vanzetto, H. (2012). TLA + proofs. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 7436 LNCS, pp. 147–154). https://doi.org/10.1007/978-3-642-32759-9_14

[10] Davis, J. A., Clark, M., Cofer, D., Fifarek, A., Hinchman, J., Hoffman, J., … Wagner, L. (2013). Study on the barriers to the industrial adoption of formal methods. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). https://doi.org/10.1007/978-3-642-41010-9_5

[11] Dongmo, C., & Van der Poll, J. A., (2016). Formalising non-functional requirements embedded in user requirements notation (urn) models. University of south africa.

[12] Fix, L. (2008). Fifteen years of formal property verification in intel. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 5000 LNCS, pp. 139–144). https://doi.org/10.1007/978-3-540-69850-0_8

[13] Gao, J., Zhang, L., & Wang, Z. (2008). Decision support in procuring requirements for ERP software. In Proceedings of the 9th International Conference for Young Computer Scientists, ICYCS 2008 (pp. 1126–1131). https://doi.org/10.1109/ICYCS.2008.158

[14] George, V. & Vaughn, R. (2003). Application of lightweight formal methods in requirement engineering. Crosstalk: *The Journal of Defence Software Engineering.*

[15] Gibbons, J., & Oliveira, J. N. (2009). Teaching Formal Methods [electronic resource]: Second International Conference, TFM 2009,

ISBN: 978-1-63248-169-6 DOI : 10.15224/978-1-63248-169-6-12

Eindhoven, The Netherlands, November 2-6, 2009. Proceedings / edited by Jeremy Gibbons, JosÃ© Nuno Oliveira. Springer eBooks.

[16] Grabski, S. V., Leech, S. A., & Schmidt, P. J. (2011). A Review of ERP Research: A Future Agenda for Accounting Information Systems. Journal of Information Systems. https://doi.org/10.2308/jis.2011.25.1.37

[17] Hussain, S., Dunne, P., & Rasool, G. (2013). Formal specification of security properties using Z notation. Research Journal of Applied Sciences, Engineering and Technology.

[18] IIBA. (2015). A Guide to the Business Analysis Body of Knowledge (BABOK). International Institute of Business Analysis, Toronto, Ontario, Canada. https://doi.org/10.1017/CBO9781107415324.004

[19] Ilić, D. (2007). Deriving formal specifications from informal requirements. In Proceedings - International Computer Software and Applications Conference. https://doi.org/10.1109/COMPSAC.2007.104

[20] Jaspan, C., Keeling, M., Maccherone, L., Zenarosa, G. L., & Shaw, M. (2009). Software mythbusters explore formal methods. *IEEE Software*. https://doi.org/10.1109/MS.2009.188

[21] Kefalas, P., Eleftherakis, G, A, S. (2003). Developing Tools For Formal Methods. *In Proceedings of the 9th Panhellenic Coneference in Informatics*. http://doi.org/http://dx.doi.org.ezaccess.libraries.psu.edu/10.3149/jms.1403.379

[22] Kilic, H. S., Zaim, S., & Delen, D. (2015). Selecting "the best" ERP system for SMEs using a combination of ANP and PROMETHEE methods. Expert Systems with Applications, 42(5), 2343–2352. https://doi.org/10.1016/j.eswa.2014.10.034

[23] Kneuper, R. (1997). Limits of formal methods. Formal Aspects of Computing, 9(4), 379–394. https://doi.org/10.1007/BF01211297

[24] Li, F.-L., Horkoff, J., Borgida, A., Guizzardi, G., Liu, L., & Mylopoulos, J. (2015). From Stakeholder Requirements to Formal Specifications Through Refinement. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 9013, pp. 164–180). https://doi.org/10.1007/978-3-319-16101-3_11

[25] Malik, P., & Utting, M. (2005). CZT: A Framework for Z Tools. In ZB 2005: Formal Specification and Development in Z and B 2005: Formal Specification and Development in Z and B. https://doi.org/10.1007/11415787_5

[26] Markus, M. L., & Tanis, C. (2000). The Enterprise System Experience — From Adoption to Success. Framing the Domains of IT Management: Projecting the Future Through the Past, 173–207. https://doi.org/10.1145/332051.332068

[27] Nwankpa, J. K. (2015). ERP system usage and benefit: A model of antecedents and outcomes. *Computers in Human Behavior*. https://doi.org/10.1016/j.chb.2014.12.019

[28] Palmquist, M. S., Lapham, M. A., Miller, S., Chick, T., & Ozkaya, I. (2013). Parallel Worlds: Agile and Waterfall Differences and Similarities. SEI, Carnegie Mellon University. https://doi.org/CMU/SEI-2013-TN-021

[29] Pressman, R. S. (2009). Software Engineering A Practitioner's Approach 7th Ed - Roger S. Pressman. Software Engineering A Practitioner's Approach 7th Ed - Roger S. Pressman. https://doi.org/10.1017/CBO9781107415324.004

[30] Preuß, S., e, N. A., Gerber, C., & Hanisch, H. M. (2011). Virtual start-up of plants using formal methods. *International Journal of Computer Applications in Technology*. https://doi.org/10.1504/IJCAT.2011.045401

[31] Royce, W. (1970). Managing the Development of Large Software Systems. Proceedings, IEEE WESCON, (August), 1–9.

[32] Seo, G. (2013). Challenges in Implementing Enterprise Resource Planning (ERP) System in Large Organizations: Similarities and Differences Between Corporate and University Environment. *MIT SLOAN School of Management*. https://doi.org/857768973

[33] Shehab, E. M., Sharp, M. W., Supramaniam, L., & Spedding, T. A. (2012). Enterprise resource planning. Business Process Management Journal, 10(4), 359–386.https://doi.org/10.1108/14637150410548056

[34] Sommerville, I. (2016). Software Engineering. Software Engineering. https://doi.org/10.1111/j.1365-2362.2005.01463.x

[35] Spichkova, M. (2012). Human factors of formal methods. In Proceedings of the IADIS International Conference Interfaces and Human Computer Interaction 2012, IHCI 2012, Proceedings of the IADIS International Conference Game and Entertainment Technologies 2012.

[36] Spivey, J. (1992). The Z notation. Prentice Hall International (UK) Ltd. Retrieved from http://www.rose-hulman.edu/class/se/csse373/current/Resources/zrm.pdf

[37] Srihasha, A. V, & Reddy, A. R. M. (2015). Modest Formalization of Software Design Patterns. International Journal of Latest Research in Engineering and Technology.

[38] Steyn, P. S. and Van der Poll, J. A. "Validating Reasoning Heuristics Using Next-Generation Theorem Provers". In The 5th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems (MSVVEIS), pp. 43 – 52. Funchal, Madeira, Portugal, June 2007. ISBN 978-972-8865-95-5.

[39] Suryalena. (2013). Enterprise resource planning (erp). Jurnal Aplikasi Bisnis, 3, 145–154.

[40] Toyn, I., & McDermid, J. A. (1995). CADi: An architecture for Z tools and its implementation. Software: Practice and Experience. https://doi.org/10.1002/spe.4380250306

[41] Tretmans, J., & Belinfante, A. (1999). Automatic Testing with Formal Methods. Analysis.

[42] Van der Poll, J. A. (2010). Formal Methods in software Development: A Road Less Travelled. South African Computer Journal.

[43] Van der Poll, J. A., & Kotze', P. (2005). Enhancing the Established Strategy for Constructing a Z Specification. School of Computing,University of South Africa, 0003, (35), 118–131.

[44] Wassyng A., Lawford M. (2003) Lessons Learned from a Successful Implementation of Formal Methods in an Industrial Project. In: Araki K., Gnesi S., Mandrioli D. (eds) FME 2003: Formal Methods. FME 2003. Lecture Notes in Computer Science, vol 2805. Springer, Berlin, Heidelberg

[45] Woodcock, J. I. M., Larsen, P. G., Bicarregui, J., & Fitzgerald, J. (2009). Formal Methods: Practice and Experience. ACM Computing Surveys, 41(4), 40. https://doi.org/10.1145/1592434.1592436

[46] Xilinx. (2005). Xcell Journal, Issue 55 Fourth Quarter 2005. Technology.

Proc. of the Eighth Intl. Conf. on Advances in Computing, Communication and Information Technology - CCIT 2019

ISBN: 978-1-63248-169-6 DOI : 10.15224/978-1-63248-169-6-12

About the Authors:

Aifheli Nemathaga is a qualified software engineer at a South African software company. His research interests are in Formal Methods for correct software development.

John Andrew van der Poll holds a PhD in Computer Science from the University of South Africa (Unisa). He is a full professor at the Graduate School of Business Leadership (SBL) and his research interests are in the construction of highly dependable software for Business ICTs.