Understanding the Efficacy of Graphical Formal Methods-Empirical Assessments

Rozilawati Razali

Centre of Software Technology and Management, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia

Abstract: The use of mathematical notations in formal methods increases a system model’s precision and consistency. The notations however are regarded as being difficult to comprehend due to unfamiliar symbols and interpretation rules that are not apparent to practitioners. Graphical methods use visual or graphical notations to present system elements. They therefore are perceived as more accessible. Nevertheless, such notations cannot be verified systematically to ensure a model’s correctness. Graphical formal methods that contain formal and graphical notations could perhaps produce a model that is not only accurate and consistent but also accessible to practitioners. This paper discusses the efficacy of graphical formal methods in specifying system requirements. The discussion is based on a series of empirical assessments conducted on such methods together with some theoretical explanation. The discussion provides practitioners with some understanding of the strengths, weaknesses, opportunities and threats of methods that integrate graphical and formal approaches.

Key words: Formal methods • Formal notations • Graphical notations • Empirical assessments

INTRODUCTION

Formal methods have been proposed as a part of software development lifecycle for many years. They have been applied in a wide variety of settings particularly in the development of complex and critical systems [1], where the issue of safety and security is the main priority. Although formal methods encompass the entire process of system development, they are widely used in the specification development.

A formal specification is a system description expressed in a notation whose vocabulary, syntax and semantics are formally defined using mathematical constructs. The specification is a useful artefact as it helps improving the quality of the later product [2]. The formality imposed by the notation enables the early detection of specification errors, which are expensive to correct if they remain undetected until later stages of software development process [3].

Despite many asserted advantages that formal methods and their artefacts could offer, there is still much debate on the practicality of the methods [2, 4, 5]. One of the major concerns with formal methods is the ability of practitioners to overcome the mathematical barriers in a formal specification. The mathematical notation used in the formal specification is always perceived as difficult to read and understand [6-8]. Therefore, it is not surprising that the successful use of formal methods in research laboratories has had little real impact on industrial practice for day-to-day software development [9].

On the other hand, modelling methods that employ graphical notation mainly use abstract visual representation for specifying software requirements. The notation possesses some modelling rules and support refinement activities. It however cannot be verified systematically to ensure the consistency and accuracy of the specification. Even if it is possible to confirm the interconnection between entities and perform syntactic checking, it is almost impossible to ensure each successive refinement’s correctness.

Previous studies have shown that graphical notation has significant advantages for certain tasks particularly software comprehension [10, 11]. The notation is in some sense analogous to the world that it represents and thus is more intuitive [12]. Researchers have though yet to agree the superiority of graphical notation over textual
One reason is that the underlying factors that contribute to the superiority of graphical notation are not well understood [14]. Moreover, a purely graphical notation is not as expressive as a textual notation. It is better to take the view that a graphical notation plays an important role in software comprehension as a companion to a textual notation.

In software engineering, it is important to choose a usable notation for software specification. The notation used determines the comprehensibility of a specification, which indeed is an essential communication mechanism among different stakeholders. Specifications must be comprehensible to stakeholders with diverse backgrounds and expertise. It seems that the notation which is easier to learn and use is more preferable among practitioners. Nevertheless, the accuracy of the specification also needs to be preserved. By combining both graphical and formal notations could perhaps produce a specification that is not only accessible to practitioners but also correct and consistent.

Unfortunately, the choice of notation for a particular project often reflects the experience or preferences of the development team more than an objective consideration of possible alternatives [15]. Lack of empirical evidence to support theories, models and decisions in software engineering is one of the reasons [16]. Unless the specific factors that cause a software technology to be more or less effective are understood, its adoption will continue to be a random act. Researchers therefore must learn to produce evidence that is useful for practitioners. Software technology must be evaluated empirically and rigorously to determine any significant and quantifiable improvement [17]. It is essential to seek some evidence of a technology’s likely efficacy when used under certain conditions.

This paper is intended to discuss the efficacy of graphical formal methods for specifying software requirements from the perspective of What, Why, Who, Where, When and How aspects. The discussion is based on a series of empirical assessments conducted on such methods together with some theoretical explanation found in the literature. The paper focuses on one instance of graphical formal methods that combines Unified Modelling Language - UML (graphical) and B (formal) as the object of discussion. The findings can however be generalised to any modelling methods that incorporate both approaches to produce a precise and accessible model. It provides practitioners with some understanding of the potentials and limitations of graphical formal methods. Practitioners could use the findings as the basis to plan, design and use graphical modelling methods effectively.

The paper is organised as follows: The following section provides background information on integrating graphical and formal notations as well as empirical assessments on modeling notations. It is then followed by discussion about issues and recommendation for graphical formal modeling methods. Later, some potential future work on such methods is described. Finally, the last section summarises the content of the paper.

**Background**

**Integrating Graphical and Formal Notations:** Formal methods have been regarded as one approach towards high quality software. Formal notations used in formal methods such as in the B method contain textual mathematical constructs. They have the precision, which enables them to be rigorously verified so that a correct and consistent model can be produced. The notations however are generally difficult for untrained users to understand because they have a large numbers of different symbols that represent complex operations [18]. Even with training, several studies have found that formal notations are not easily readable and understandable [6, 8].

Graphical notations such as UML mainly use graphical symbols to represent structural and dynamic aspects of a system. Graphical symbols may be effective for communicating ideas but they are not as expressive as textual constructs. In particular, one cannot use graphical symbols alone to specify system constraints. As the expressive power of a representation is essential for model completeness [19], a model that uses graphical symbols may be incomplete. Moreover, the lack of formality in graphical symbols causes model verification to be difficult, if not impossible.

A study has indicated that the presence of graphical symbols together with a formal textual notation could improve the notation’s readability [9]. This may suggest that by having UML diagrams with the formal notation of B could also lead to similar results. In essence, integrating both notations may address the lack of formality in UML while improving the accessibility of B. Furthermore, such integration could also benefit from the B’s strong industrial supporting tools and the wide acceptance of UML in industry.
Fig 1: The transformation of a graphical model to a formal model

Realising the potential of such integration, the research community has attempted to establish links between UML and B [20-25]. These studies mainly investigate the ideas of transforming a UML model to a B model. However, there are also studies that investigate the ideas of reverse transformation, that is, from a B model to a UML model [26]. Regardless of direction, these studies aim to exploit the strength of B tools while remaining in a standard industrial process based on UML. They are also motivated by the belief that UML could help B notation to be approachable to practitioners.

The discussion in this paper is based on an instance of methods that combines the use of a graphical notation, UML [27] and a formal notation, B/Event-B [28, 29]. The method is called UML-B [30]. UML-B uses UML’s Package, Class and Statechart diagrams as the graphical representation of its model. In general, the transformation of a UML model to B model is achieved through formal semantics and translator as shown in Figure 1 below. In the case of UML-B, the UML graphical notation is equipped with formally defined semantics by using an integrated and action language called µB or microB. µB is based on the B’s AMN notation. A translator called U2B translates a UML-B model to a textual B model so that B tools can be executed to verify the model.

Empirical Assessment of Modelling Notations:

Although the essence of having high quality software has been realised, software disasters and user dissatisfaction are still prevalent in the industry [31]. Increasingly large amounts of resources are being allocated to software projects, yet software quality is no better today than it was some time ago. In fact, delivering software on schedule and within the development budget is still a problem. This issue is partly due to the lack of measurement programmes that explore the efficacy of the available software technology used in software development. Software experts seem to be more interested in inventing technology rather than evaluating it. The situation forces practitioners to continue using technology, whose effectiveness remains mostly un-investigated.

It is vital to assess various software methods and tools to understand how they can help produce high quality software within the specified budget and schedule. Measurement provides an ideal mechanism for assessing software technology [32]. Through measurement, the appropriateness of a technology can be assessed in terms of its effectiveness and usability. Conducting empirical assessments is how measurement can be applied to achieve the objective. The findings of one single investigation are indeed insufficient to explain a phenomenon. Therefore, several assessments that investigate the phenomenon from different perspectives using different complementary approaches are needed.

A number of empirical studies have investigated the efficacy of different notations for model interpretation and model creation tasks. For model interpretation, there are studies that compared notations of different technologies such as Data Flow Diagram (DFD) versus Object-Oriented Technology (OO) [33] (Agarwal et al., 1999), OO versus Extended Entity-Relationship (EER) [34], DFD versus task-oriented menus [35] and UML versus Open Modelling Language (OML) [36]. Several studies compared internal properties or structures of one particular notation such as optional versus mandatory properties of Entity-Relationship Diagram (ERD) [37], different decomposition [38] and types of UML diagrams [39]. There are also studies that assessed a notation presented using different modes such as OO using narration versus animation [40]. These studies mainly used comprehension as the measure of interest, which was determined by the accuracy of the responses given by participants who read the presented models.

In terms of model creation, there are comparisons of notations and methods between different technologies. Some examples include DFD versus OO [41], DFD versus OO versus ERD [42], DFD and Integrated Definition Method (IDEF0) [43], Flowchart and Program Design Language (PDL) [44], Relational Data Model (RDM) versus EER [45] and Logical Data Structure (LDS) versus Relational Data Structures [46]. The most common measures of interest used in the studies include model correctness, ease of use and ease of learning.
Most of the studies mentioned above assessed the notations and methods that are graphical. The representations mainly consist of graphical symbols with some “non-formal” textual notations. The textual notations do not contain mathematical constructs and are generally based on natural language. On the other hand, there are studies that explored the readability of formal notations for model interpretation task. For example, comparisons of formal notations with informal specifications [8] and code [47] and comparisons of formal notations using different presentation modes [9] and structures [6].

Studies that investigated the efficacy of notations that combine formal (textual mathematical constructs) and graphical notations empirically are almost non-existent. One study assessed the use of Object Constraint Language (OCL) with UML diagrams [48]. It explored the use of UML diagrams with OCL as compared to UML diagrams. The study indicates that using dual notations together is better than one notation. It seems to suggest using UML and OCL (graphical and formal notations) rather than UML alone (graphical notation), particularly when practitioners are properly trained and mentored.

This paper is particularly referred to empirical assessments made on a method that combines UML and B. The assessments comprise a series of controlled experiments and surveys. The controlled experiments evaluated the comprehensibility of the model produced by the method from practitioners’ perspective for software validation and maintenance or model interpretation purposes [49, 50]. On the other hand, the surveys were intended to evaluate the usability of the method [51-53] and the supporting tools from developers’ perspective for model creation purposes [54]. In addition to the empirical assessments, a theoretical analysis was also conducted on the method to understand its efficacy. The analysis explored several theories from the Information Science, Cognitive Science and Educational Psychology disciplines to explain the phenomenon. This provides software practitioners with some understanding of the strengths, weaknesses, opportunities and threats of the method. The detailed discussion of the findings is elaborated in the following section.

**DISCUSSION**

In this section, the findings of empirical and theoretical assessments are gathered to explain the efficacy of graphical formal methods such as UML-B. Although UML-B is used as the object of discussion, it is believed that the aspects discussed are applicable to other methods that have motivation and characteristics similar to UML-B. In particular, methods that incorporate a graphical notation into a formal notation to produce a model that is not only precise and consistent but also accessible. The discussion is organised into What, Why, Who, Where, When and How aspects as follows.

**What:** “Graphical formal methods” or GFM is the term used in the paper to mean development methods that combine the use of graphical and formal notations in creating software artefacts. The artefacts originate as early as the system specification stage. Graphical notations are representations that provide a set of visual symbols to represent specific system elements. Some examples of graphical notations include Entity-Relationship Diagram (ERD), Data-Flow Diagram (DFD), Unified Modelling Language (UML) and Open Modelling Language (OML). On the other hand, formal notations such as Z and B use textual mathematical symbols and interpretation to describe a system. The notations have rules for determining the grammatical well-formedness of sentences, rules for interpreting sentences in a precise and meaningful way and rules for inferring useful information from a specification [55]. Besides the format, formal notations are essentially different from graphical notations because the former define properly the intended properties and behaviour of system elements.

Theoretically, GFM encompass any possible combination of graphical and formal notations. For the integration to be useful however, there are some compatibility criteria that the participating notations have to meet. First, even if not entirely, there are some basic principles in both notations that fit together. For instance, if the graphical notation is best suited for data flow representation, the formal notation should support the characterisation of data entities and transitions. Similarly, if the graphical notation supports object-oriented development, the formal notation should possess certain features of objects. In the case of UML-B for example, B has many features similar to UML such as encapsulation of operations with associated state variables and identification of entities with specific relationships. Second, the integration must be complementary where the notations appear as one joined representation rather than two different ways of presenting the same information element. Each notation highlights different types of information that it suits best. The graphical notation can be used to illustrate elements.
that are readily interpreted such as grouping of entities, relationships between entities and transitions between states. The formal notation is used to describe literally the properties, behaviours and constraints of the entities that cannot be illustrated by the graphical notations. The integration of the two qualitatively different notations is enabled through specific rules that define the complementary roles of each notation.

The primary objective of GFM is to promote the use of formal or mathematically-based methods in software development. While being able to produce a reliable system [4, 5], barriers towards formal methods are recognised [6, 7]. The aim of GFM is to enable formal methods to become more accessible. UML-B for instance, is intended to overcome barriers towards B so that it can be widely used in industry [30]. Therefore, the approach taken by GFM differs from the ordinary view of graphical and textual representations integration such as UML and its formal syntax; Object Constraint Language (OCL). OCL is designed as an annotation notation for UML diagrams so that the diagrams can be more expressive and precise. The intention is primarily for the presentation and interpretation of information where OCL (formal notation) augments UML diagrams (graphical notation). In contrast, GFM go beyond just that. The aim is to produce a formal model that is accurate and precise through thorough validation and verification activity. The activity however is easier to accomplish only if users understand their own models well. The use of graphical notation helps in achieving the aim by improving a formal model’s accessibility among its users.

GFM are geared towards facilitating verification activity, which is generally assisted by special tools. Ideally, such tools should be able to discharge the inaccuracies and inconsistencies of a model automatically. As the tools perform the verification activity based on predefined rules, hypotheses and theorems of proving, there are limitations on the way they can prove a model. To enable the tools to discharge errors as efficiently as possible, GFM manipulate the participating notations into forms that are easier for proving. Due to this reason, there are differences in styles when the notations are used together and when they are used individually. This particularly affects the graphical notation as the verification tools are designed to interpret the formal notation. For instance, to appropriately represent the mapping of sets in B that can easily be verified, the use of class multiplication and association in UML-B is quite different from the conventional UML. This claim is supported by the results of the surveys [51-53].

The first step in using a formal method is to employ its formal notation in specifications. Therefore, the role of GFM begins from the specification and design stage. The sources or inputs for conducting specification and design tasks using the methods are user requirements of a system. The intermediate products of the methods are formal specifications or models, which act as the means of communication and reference among stakeholders. If the formal method is employed extensively, the specifications can generate executable code and test cases for later stages.

Why: GFM aims at promoting the individual strengths of graphical and formal notations and complementing one notation’s weaknesses by the strengths of the other. Formal notations have the ability to increase a model’s precision and consistency, as special-purpose tools can systematically verify them using specific proof theory. The notations however are generally difficult to comprehend because of their mathematical symbols and rules of interpretation [8]. Graphical notations employ visual symbols. The visual symbols, whose concepts are similar to objects in the real world, are mainly intuitive [12]. Graphical notations however are not as expressive as formal notations as visual symbols cannot illustrate system properties completely [56]. Moreover, visual symbols cannot be verified systematically to confirm the accuracy and consistency of a model.

GFM are capable of improving formal or mathematically-based software development. In particular, the methods promote formal models’ comprehensibility while yet offering greater confidence that correct models are developed. The methods could ensure an effective validation task for formal models by getting stakeholders more involved in the process. Stakeholders’ involvement during the validation task is critical for the successful software development [31]. The visualisation aspect of GFM provided by the graphical notation could encourage the participation of stakeholders as it improves the readability of formal models. With reasonable hours of formal training, stakeholders should be able to interpret the models. The readability of the models could also facilitate efficient maintenance where maintainers can easily grasp the required information prior to maintenance tasks. These claims are particularly supported by the results of the controlled experiments [48-50].

Who: GFM are appropriate for users who favour a natural approach of formal modelling. Specifically, the methods mainly benefit new formal method users who are already
accustomed to using visual modelling techniques. To these users, the use of graphical symbols together with textual mathematical symbols at abstraction level is regarded as more approachable than using the mathematical symbols entirely. This claim is supported by the results of the surveys [51-53].

Combining graphical and formal notations in GFM allow the strengths of both notations to be reinforced. But more importantly, it permits one notation’s limitations to be compensated by the other. Such methods need users to be aware of the strengths and weaknesses of the participating notations. The awareness enables users to understand the motivation behind the integration, appreciate the effort and form reasonable expectation.

The integration of graphical and formal notations in GFM involves the use of two individual notations, which are qualitatively different in the way they present information. There are significant differences on how textual and graphical representations illustrate system elements, as mentioned earlier. Moreover, the individual notations are normally accompanied by certain methodology or rules that guide their application. For instance, B method defines the ways B notation should be employed in specifications to enable verification and refinements in later stages. In contrast, UML encourages system development through visualisation using different perspectives. GFM combine not only the individual characteristics of the notations but also their specific strategies of supporting system development. Users of such methods therefore should have the capacity to grasp the underlying principles and methodology of both participating notations.

GFM entail some adjustments to be made on the participating notations. For instance in UML-B, a set of appropriate B/Event-B syntax is attached to UML diagrams for specifying textual constraints and actions. Despite being a mathematically-like notation, the syntax has object-oriented elements to suit UML’s style of modelling. The UML diagrams are matched to the B/Event-B syntax through specific rules of integration. The integration rules cause the notations to operate differently together from how they would individually. Rather than thinking of the notations independently, users of GFM thus need to understand how both notations work together under the rules. As each notation represents one aspect of a system that is not represented by the other notation, the roles that each notation plays in the integration should also be understood.

The models of GFM are likely to contain several elements of information as one representation illustrates one aspect of a system. This requires the models to have scattered parts that should be related and interpreted together to form a meaning. For example, a UML-B model has UML-like diagrams and B/Event-B syntax. In fact currently, there are several types of diagrams in UML-B that present system information from several different perspectives. The information presented in one diagram should be related with the one presented in the other diagrams. Users with high spatial ability, who are able to conceptualise the spatial relations between different types of representation and parts, are believed to be at an advantage when dealing with such models.

Successful technology transfer requires not only a new idea but also a receptive audience [17]. From the marketing perspective, the audience can be divided into two categories, namely Early Market and Main-stream Market [57]. The Early market audience requires little evidence of the efficacy of a new technology. They therefore comprise people who are eager to try any new technology other than they use normally. They also include people who are driven by their perception of a new technology’s potential benefits, which is built from others’ success stories. This audience is willing to take great leaps and welcomes major changes to the current practices. In contrast, the Main-stream Market audience requires more evidence. They are more cautious, careful and skeptical people who will only adopt a new technology if it is necessary. They prefer to make minor changes where the new technology enhances rather than replaces the current practices. In the context of GFM, they are more likely to be accepted by the Early Market audience. This is because the technology and idea are very new, which require more evidence to be gathered. Besides, there are a number of requirements that need to be met for adopting the methods, which will be explained later.

Where: GFM combine the use of graphical and formal notations in a model, which is later transformed to a formal model that can be verified and executed. As this involves interaction between different representations and transition between several stages, manual operation of such a process can be complicated and troublesome. GFM therefore should be highly automated and equipped with strong supporting tools.
Supporting tools are of critical importance. The tools encompass the internal devices that support the modelling activity and the external devices that prepare users with the necessary knowledge and expertise. The internal devices include the editor in which the model is developed, a translator that transforms the model to a formal model completely and verification tools that prove the accuracy and consistency of the formal model. The external devices include comprehensive documentation about the methods’ principles and the features supported by the tools and user training. The tools should assist users to capture the essence of the methods and use them correctly. Complexity of dealing with multiple notations and modelling elements should be reduced through a simple and user-friendly modelling environment. The tools are also expected to operate as efficiently as possible. These claims are supported by the results of the surveys [51-53]. Some features that are necessary for the tools to become usable have also been discovered [54].

When: GFM are indeed formal or mathematically-based methods. Although GFM use graphical symbols together with textual mathematical symbols, they gradually produce formal (mathematical) models. Formal methods in general emphasise the need of having reliable systems. Systems developed using formal methods are believed to be safe and predictable. This is because the specified system properties and behaviours are proven systematically to be accurate and consistent under all known conditions.

Reliable systems assured by formal methods come with a price. Formal development can be exhaustive and resource extensive. System properties and behaviours have to be defined unambiguously and cross-checked across several levels of abstractions and refinements. The process is interactive where system elements are added, removed and manipulated during verification activity based on predefined hypotheses. Although verification tools should be able to generate hypotheses and prove them automatically, there are still occasions where users have to intervene. This is particularly true when system requirements are incomplete or contradict one and another. Users should be aware of various possible conditions and understand how the system behaves under those conditions. This is to enable them to formulate hypotheses for the verification tools to base on.

Formal development is difficult for users to master instantly. The verification task particularly is time consuming and troublesome as users need to understand not only the system to be developed but also how to apply the methods correctly. Formal notation in itself is not easy to grasp. Users need to understand the meaning and properties of the notation and how it may be manipulated. Users thus need strong support from the environment. It has been suggested that users should have regular access to experts during formal development to ensure the successful use of the methods [2, 5]. This means organisations may need to allocate a budget to hire experts for projects using formal methods. Effective and efficient tools that support the development should also be available and usable. Moreover, users should be prepared with the necessary knowledge prior to using the methods. Some time has to be allocated for users to acquire expertise and fluency to express new problems, solutions and proofs using the methods. These requirements have been discovered from the surveys [51-54].

The development process using GFM can be different from other process models. Such methods emphasise the need of having an accurate and consistent specification of the system through thorough validation and verification activity. The assumption is that whenever the front-end process has been carefully handled, the end product can be assured to be reliable. The formality imposed at early stages can highlight issues that might not be discovered at later stages. Therefore, the early development stages using the methods can be longer where several levels of activities have to be executed before the artefact can move to the next stage. In fact, if the front-end process is executed properly as intended, the later stages such as coding and testing can be shortened. In the case of UML-B for example, the generated and verified B/Event-B model can be translated automatically into executable code if it has been refined at a sufficiently low level. While integration testing may still be required, the code may not need extensive unit testing as its internal accuracy has been systematically verified.

The above arguments indicate that GFM do require high operational cost and investments. Similar to conventional formal methods, GFM are believed to be particularly suitable for the development of critical systems where system safety and reliability is the main priority. Such systems cannot afford to fail. The requirement of having safe-guaranteed systems that reliably operate undermines the high cost of its development. Organisations that opt into using GFM should be willing to make some adjustments to their current process model to suit the methods’ way of operating. The allocation and distribution of resources
may also need rearrangement. Organisations are advised to have a cost estimate and some idea of anticipated costs before embarking on such methods.

**How:** System development using GFM consists of several phases. System properties and behaviours are firstly illustrated using graphical notations and formal (mathematical) notations. Both notations are used together to create a model of the system based on predefined rules of integration. In this first phase, the produced model is suitable for stakeholders such as clients who need to only interpret the model for user requirements validation purposes.

One important objective of GFM is to assure the accuracy and consistency of the system to be developed. To achieve that, validation and verification activity takes place as early as the specification stage. Despite being formal, the initial model produced by the methods contains graphical symbols that might not be verified systematically using tools. The methods thus have a mechanism to transform the graphical formal model to a textual formal model that is viable for verification. This is achieved by having a translator that transforms each graphical symbol to the equivalent textual formal representation. The transformation is based on the mappings defined in the methods’ principles and rules of integration.

Automated tools support verification activity in GFM. The tools interpret the model and verify its accuracy based on predefined hypotheses and proving theorems available in the proof library. Users are informed of any detected syntax errors and semantics that cannot be proven. Users should make the corrections on the model and introduce new hypotheses to the proof library for the tools to discharge the errors. Verification tools are generally sensitive and use specific techniques to discharge errors efficiently. Users should be accustomed to the techniques. The tools could assist users in understanding the model and verification task by having an animation facility. The facility provides users with some kind of visualisation that shows the behaviours of the model under the specified conditions. A survey has discovered a set of basic features that are necessary for the tools to become usable [54]. The model that has been verified by the tools is considered as accurate and consistent, which may be translated to code.

In general, there are two technical aspects that are important in GFM, namely environment and notation. The role of environment has been discussed in earlier paragraphs. Notation is vital as it is the starting point of the whole idea of integration. Most software systems are large and complex. No single notation will address all aspects of a complex system. The participating graphical and formal notations should be suitable for the integration where they complement each other to fit the system that they are meant to describe. The suitability of the notation used in GFM can be evaluated through its expressiveness and effectiveness. Expressiveness refers to the ability of the notation to represent the intended information whereas effectiveness concerns the efficacy of the notation as a means of representing information. Notation expressiveness and effectiveness can be assessed empirically based on the tasks that the notation uses. As the notation is used to describe a problem domain in a model, two major tasks include model interpretation and model creation [40]. Model interpretation involves understanding of the domain being represented by interpreting the model. Model creation involves communicating one’s understanding of the domain by presenting it in a model.

It has been suggested that the expressiveness of a notation should be considered from a theoretical perspective [58]. The theoretical consideration can be used to guide empirical investigations and to suggest ways for creating more effective notations. Ontological evaluation [59] is one such assessment. Ontological evaluation concerns whether the notation has been designed to allow users to create a model that provides clear descriptions of the domain being represented. The evaluation is based on ontology, which is a set of beliefs of what might exist and happen in the domain. Four types of ontological evaluation on notation have been proposed [60], which include Construct Deficit, Construct Overload, Construct Redundancy and Construct Excess. The evaluation can be extensive depending on the scope of the investigation. Table 1 below provides an example of brief ontological evaluation of UML-B. The intention is to show the applicability of ontological evaluation for identifying some limitations of UML-B’s notation.
There are two sources of mental or cognitive load, namely intrinsic and extrinsic [61]. Intrinsic cognitive load concerns the intellectual complexity of the information whereas extraneous cognitive load refers to how the information is presented. Intrinsic and extraneous cognitive load together contribute to the total cognitive load involved in understanding. Intrinsic cognitive load is determined by the degree to which information elements presented in a material interact [61]. Some information can be processed sequentially without having to relate its individual elements with one and another. The low interactivity of information elements causes low intrinsic cognitive load in the working memory. In contrast, some information requires its elements to be processed simultaneously rather than sequentially. The information is highly interactive, as the elements involved cannot be separated as autonomous entities. This causes high intrinsic cognitive load in the working memory. Extraneous cognitive load is determined by the method used to present the information. Each method of presentation varies in extraneous cognitive load. One example of extraneous cognitive load is when the information presented in a material is physically separated. This requires users to use cognitive resources to search and match the related information to form a meaning. This causes high extraneous cognitive load in the working memory as the process of searching and matching information interfere with the process of understanding. The phenomenon is called the Split-attention effect [62]. It has been suggested that understanding could be enhanced if the related information is physically integrated. This could reduce the use of cognitive resources for tasks other than the understanding itself.

Formal (mathematical) notations in general are likely to cause high intrinsic cognitive load. This is because they involve concurrent interactions between syntactical and semantic elements. Each syntax or symbol of a formal notation has a specific meaning. When a set of syntax is used together, the individual meanings and constraints have to be integrated as joined elements that represent a single interpretation. In the case of GFM, its model contains formal syntax. Thus, the model can be regarded as having high intrinsic cognitive load similar to its counterparts, formal models. However, the results of the controlled experiments [49, 50] suggested that a graphical formal model is more comprehensible than a formal model.

Based on the cognitive load theory, the total cognitive load in working memory is a function of both intrinsic and extraneous cognitive loads. A graphical formal model differs from a formal model mainly on the method of presentation. Specifically, a graphical formal model contains graphical and textual mathematical symbols whereas a formal model has only the latter. Considering the fact that both models contain similar textual mathematical symbols (high intrinsic cognitive load), it seems that the use of graphical symbols contributes to the difference in comprehension. The total cognitive load of a graphical formal model is lower than its counterparts due to its low extraneous cognitive load. This indicates that the understanding of a high intrinsic cognitive load notation can be improved if it has a low extraneous cognitive load. For instance, the comprehensibility of a formal notation can be better if it is presented in a way that is easy to perceive.

Even though a graphical formal model causes little extraneous cognitive load, it is worth knowing which parts of the model that cause it. A graphical formal model has physically separated parts, which contain information that has to be mentally integrated. Users have to search and match the separated information to understand the presented problem domain. This particularly happens between different diagrams, diagrams and their textual specifications and between textual specifications of different diagrams. For instance, a UML-B model requires switching between Class and Statechart diagrams to link system properties with behaviours and to understand the interaction among classes’ behaviours. As one screen cannot display all diagrams and textual specifications, the UML-B modelling environment involves multiple screens or panels that contain different information that has to be combined for understanding. This limitation has been discovered from the surveys [51-53].

Based on the theory of limited cognitive processing capacity of human memory, educational psychology research has investigated the role of rote understanding as an alternative pathway to meaningful understanding especially for novices [63]. Rote understanding is defined as learning discrete elements of information without the knowledge of the connection between separate elements. In contrast, meaningful understanding involves the learning of not only the individual elements but also the interconnection between them. The research suggests that when novices learn complex information, it would be better for them to be firstly exposed to rote understanding. This means the novices are presented with abstract and discrete information before being exposed to details. The rationale of this claim is that complex information has high intrinsic cognitive load, which involves a large degree of interactivity between
elements of information. Novices have very limited knowledge about a specific aspect. If novices are introduced into details directly, the working memory is overwhelmed with various new elements that have to be processed concurrently. Failing to digest any particular elements may jeopardise overall understanding. This also affects the development of mental models in the long-term memory, which is important for future understanding. Experts on the other hand may not experience overloaded working memory. They already possess specific knowledge, which are organised as structured mental models in the long-term memory. The mental models are ready to be triggered whenever needed. Experts are able to understand complex information much easier than novices as the burden of processing the information is mostly transferred to the long-term memory. The working memory load is reduced and it thus has the capacity to allow understanding to develop easily.

Stakeholders who are involved with GFM are believed to be mainly novices. Even though the problem domain presented by a graphical formal model may not be too alien to stakeholders such as clients or domain experts, the notation and concepts of GFM might be quite new to them. This is due to the unfamiliarity of software practitioners towards formal methods in general. Similarly, developers who adopt GFM may not be familiar with the method and problem domain. The idea of rote understanding discussed above might be applicable to GFM. UML-B for instance adopts a Top-down approach where information is presented in a hierarchical order. The top level Package diagram provides an overview of the interaction between Class and Context diagrams. The highest level of a Class diagram illustrates only the interactions between classes. To view the behaviours of a class, the Statechart diagram contained in the Class diagram has to be selected. This seems to indicate that UML-B reinforces rote understanding where the abstract illustration of a system is presented in front and the details are hidden. Only when necessary, the details are displayed. This may provide an explanation of why a graphical formal model is better than a formal model in promoting meaningful understanding of presented problem domain among new users [50]. Unlike a graphical formal model, a purely textual formal model illustrates system properties and behaviours at once. This may cause an overloaded working memory, which hinder novices from absorbing new complex information.

Educational psychology research has recognised the potential of dual modality format as an effective way of presenting information [64]. The cognitive theory suggests that the working memory is composed of multiple channels, which process different types of information independently with little interference [65]. The channels include a visual system for dealing with visual images and an auditory system for processing verbal information. It seems that GFM are more effective in portraying information than methods that use single notation. One reason is that GFM utilises the capacity of the working memory efficiently [50].

Cognitive and educational psychology theories have offered some explanation of the effectiveness of notation and presentation method used in GFM. The evidence obtained from the empirical assessments supports the theories. Dual modality format such as used in GFM seems to be beneficial when the information to be presented is complex. Complex information involves high interactive elements. The use of graphical symbols and textual mathematical symbols can be effective in promoting understanding especially among novices. This is particularly true if the notations illustrate qualitatively different elements in a hierarchical order with minimal use of cognitive resources to search and match the related information. These findings and claims may be preliminary, which entail further investigation. However, they act as a starting point for practitioners to understand the efficacy of GFM and how it can be improved.

In general, the discussion indicates that GFM have the potential for being an approachable formal (mathematical) development technique. The methods could increase the correctness of the system to be developed not only through systematic verification but also effective validation by stakeholders. The use of graphical and textual mathematical symbols promotes better understanding than the textual symbols alone as it utilises the capacity of human memory. On the other hand, such methods could incur high investment and operational cost. They require organisations to make adjustments to their current process models and practice. Moreover, users should be equipped mentally with knowledge and expertise and physically with supporting tools. Such methods therefore are mainly appropriate for safety-critical systems, where reliability is more critical than any other concerns.

The integration of graphical and formal notations in itself is not easy to accomplish, as the notations are indeed quite different in principles and application. Any attempt to integrate the notations should be started with reasonable aims. It may however evolve over time. Rather than targeting solving various software problems, such an attempt should focus on specific system aspects.
that the notations suit best. The attempt should also consider the interaction of the notations with their targeted environments such as supporting tools and intended users.

**Future Work:** The understandability of notations is more likely to be influenced by their specific internal characteristics. It may be that the structural properties such as how the notations physically enhance the presentation of system elements in a model. If the notations could illustrate system elements in patterns that make them easy to perceive, understanding would be facilitated. The Gestalt Laws [66] for instance have outlined a set of pattern perception rules, which state graphical objects that are similar, close and move together are perceptually grouped and continuous lines are perceived more readily than contours that rapidly change direction. There are also other related perception rules that indicate objects within an enclosed region of space are perceptually grouped [67] and objects connected by continuous contours are perceived as related [68]. Moreover, object shape, colour and surface texture also play a role in facilitating understanding. For example, studies have suggested that 3D objects are easier to analyse and remember than 2D [69], which were based on the theory that humans perceive objects as composed of simple, linked, 3D solid-shape primitives [70]. Furthermore, certain graphical notations bias towards certain problem solutions [71]. Having a graphical notation per se is thus insufficient. The notation must enable the important patterns to be readily perceived. On the other hand, the degree of which the textual notation carries the meaning and the underlying interpretation rules also affect the understanding. This suggests a hypothesis that two models contain two similar integrated notations will not be informational equivalent and computational equivalent [72] although they both contain graphical and textual representations. Perhaps the integrated notations that incorporate the human pattern perception theories and fit the problem domain would be more helpful for promoting understanding.

Graphical formal models contain graphical and textual notations that complement each other to achieve the integration objectives. Although both are important, it is hypothesised that one representation contributes more to the understanding than the other. Perhaps one representation is better than the other in attracting users’ attention where users depend mostly on it for understanding. Moreover, one representation may be more influential than the other in directing how a model is understood. One way to test these hypotheses is by using eye movement tracking where the pattern of users’ attention when reading a model is recorded using an eye tracker. As eye movement provides insight into attention allocation, it is also possible to infer the underlying cognitive processes. The idea of eye movement tracking has been previously used to investigate how programmers read code [73]. It is possible to apply the idea to investigate how stakeholders read models that combine graphical and textual representations. The findings of such studies could be used to formulate strategies for improving the accessibility of the models.

The Cognitive Theory of Multimedia Learning [64] has postulated that the understanding of a material could be facilitated if it utilises the dual channels of information processing in human’s working memory, namely verbal and visual. While the visual channel mainly processes images that originated from the eyes, the verbal channel processes sounds that derived either from printed words through the eyes or spoken words through the ears. Future studies may need to explore the effectiveness of having spoken words or narrations together with the printed images to present information. For instance, it would be interesting to investigate whether or not by having narrations with diagrams could improve the comprehensibility of a graphical formal model.

In addition, future studies may need to carry out cost analysis on graphical formal methods. This includes cost-effectiveness analysis and cost-benefit analysis. The former involves the comparison of alternatives to determine the most efficient way to achieve the benefits while the latter determines whether the benefits outweigh the costs. Graphical formal methods are indeed formal methods, which mainly promise the benefits of having an accurate system through thorough validation and verification activities. Cost effectiveness analysis thus compares graphical formal methods and formal methods to determine which methods could achieve the benefits in a more efficient way. The analysis should also consider the additional benefit of graphical formal methods, which is being more approachable than the traditional formal methods. On the other hand, cost-benefit analysis investigates the relationship between the cost of using graphical formal methods and the value of the benefits that results from it. This analysis could determine whether or not it is worth using such methods at all.

Cost-effectiveness and cost-benefit analyses are likely to involve more than just one project. The analyses might investigate a large project or compare several smaller pilot projects that take different approaches to
solving the same software problem. They may also entail following up over a long period of time, to look at the long-term impact of using the methods. It is not possible to use a true experimental design to assess the effect, thus quasi-experimental designs and case studies are more likely to be adopted. Cost-effectiveness and cost-benefit analyses can be complex. Future studies that investigate this aspect should not be taken lightly.

Graphical formal methods are considered as formal, which use mathematical-based notations and rules of interpretation. Formal methods have their own strategies for developing software. The methods emphasise the need of having iterative refinement and verification processes. The strategies could be different from the common process models such as Waterfall [74] or even the latest approach such as Agile development [75]. Managers in general are reluctant to modify extensively the current development process employed in their organisations. This is because the employed process model has become a part of the organisations’ culture and therefore, any major changes on it would be difficult. Future studies may need to investigate how such integrated methods could be adopted with minimal adjustments to the current process models employed by organisations. Guidelines could be proposed where managers are provided with a number of approaches and options for effectively including the methods in the current processes and practices.

CONCLUSION

The paper has discussed the efficacy of graphical formal methods, which combine the use of graphical and textual formal notations. The discussion is based on a series of investigations from different perspectives using different empirical research approaches. The paper has provided some explanation of the strengths, weaknesses, opportunities and threats of using graphical formal methods. In addition, it also highlights some directions of future work so that better understanding of the phenomenon could be obtained. In addition, the ways of how the methods and the research could be improved have also been discussed.

ACKNOWLEDGEMENT

This work was supported by Malaysian Government and Universiti Kebangsaan Malaysia.

REFERENCES


