Goal-Oriented Model Driven Enterprise Architecture Validation Approach

Joe Essien
School of Computing and Technology,
University of West London, St Mary’s Road, Ealing, London
Joeekpeokon.essien@uwl.ac.uk

Abstract: Enterprise Architecture (EA) has been defined as the organization of a system embodied in its components, relationships to each other, environment, the principle guiding its design and evolution (IEEE, 2000). Thus an important characteristic of EA is to provide a holistic view of the enterprise visualizing the relevant aspects of the business for specific stakeholders. However, one of the many concerns of this interest has been how to deal with the complex challenges of implementing the models with the ability to validate its integrated components to ensure conformity with individual stakeholder’s motivation. To achieve this, methodologies that describe components in relation to their behavioral attributes, impact on other elements in the domain and their dependencies have been postulated. Albeit, studies show that these taxonomies do not adequately address validation requirements (Lankhorst, 2013). This article analyzes the EA concepts of ArchiMate, focusing on the business and application layers with the objective to extend motivation with tests specifications using the model-driven approach thus offer descriptive semantics for validation. The paper contributes to a better understanding on how EA models can be validated thus improve alignment with the business vision and strategy. Student Internship Program case study is used to exemplify this hypothesis.

Keywords: Goal-Oriented, Enterprise Architecture, Framework, Business Processes, Motivation, Validation, Modeling.

1 Introduction

Enterprise Architecture has been defined as consisting of coherent principles, methods, and models used in the design and realisation of organisational structure, business processes, information systems and infrastructure (Fischer et al, 2010). Good enterprise architecture provides the insight needed to balance requirements and facilitates translation from corporate strategy to daily operations (Lankhorst, 2013). The multi-dimensional interests and unstructured principles that tagged EA from inception culminated in use of heterogeneous set of approaches and modelling languages (Sessions, 2007). However, most EA practitioners recognize four facets of EA and agree that it comprises of the business, application, information, and technology perspectives (Salmons et al., 2010). Other practitioners such as Venkatraman in their review identified eight perspectives in which EA alignment can be achieved (Venkatraman et al., 2010). In addition to these eight perspectives, four other fusion perspectives are described, formed from the combination of two of the individual perspectives (Coleman and Papp, 2006). In many efforts to implement EA, and definitions that specify perspective, harmonization and alignments, the issues of validation are completely ignored or at best remain rudimentary. The positions do not consider the behavioral attributes of the model’s components as a process that should undergo test itself. In many organizations, EA patterns exist that encapsulate business concerns such as maintenance, upgrades, procurement, integration, acquisition and mergers, compliance in a regulatory environment and strategic planning (Weston et al., 2004), but literal analysis of these patterns shows many disparate architectures, understood by each stakeholder from different perspectives. The connections and dependencies that exist among these different views can be extremely complex in some cases (McGovern, 2004) requiring validation. Despite this, validation of Enterprise Architecture Framework (EAF) has been an aspect of EA that has been of enormous concern to practitioners though very little ideologies of how this can be carried out has been postulated. Most authors believe EA models are not reusable and are designed to actualize a specific goal after which it is achieved or at best used as a reference source. Thus model validation itself has not been properly defined and most definitions tend to specify the expected results of implementing an EA against predefined goals or establishing levels of maturity as the realization of EA initiatives within the organization. To tackle these phenomena, while some authors have continued to emphasize distinction between aspects of EA visualization (Salmons et al., 2010, Venkatraman et al., 2010), others stress use of maturity matrices to benchmark against as-is and to-be scenarios (Weston et al., 2004, GARTNER, 2013). This article looks at EA validation from an entirely different perspective and defines it as the use of objective evidence to confirm that a model meets the intrinsic goals defined by their requirements. To achieve this, this article delves into the intrinsic nature of the model artifacts with the aim to answer the simple question “what needs to be in a model to allow its validation?” The document describes the process of designing test specifications for an enterprise architecture model and uses the Student Internship program (SIP) as a case study to show a pragmatic application of the concept and how the specification can be extended to the model itself. Thus the paper is organized as follows; Section two presents a discussion on considerations of validation within existing popular EA frameworks. Section three introduces behavior driven modelling approach and presents characteristics of model components that allow validation and types of validation that can apply on the components. Section four defines the specifications for model validation and the data types needed for this type of validation. Section five reviews validation concepts in existing EAF with specific emphasis on TOGAF and ArchiMate. Section six presents the proposed methodology and framework extension for the EA and proposed extension to ArchiMate in order to allow the validation. Section seven presents a case study carried out to test the hypothesis using the student internship program. Section presents the implementation of the approach with case study while section nine concludes the paper and presents summary of findings and area of further research in this ambience.
2 Validation Consideration in Existing EAF

Though the practice of EA continues to evolve new techniques and collaborations, one other issue facing enterprise architects today is that none of the single practices is capable of satisfying all necessary aspects of the enterprise identified collectively (Noran 2003). Attempts to mix and match rather, has resulted in EAs with inconsistent semantics and weak ontology. Therefore issues regarding systematic validation become inconsequential as architects do not need to stringently scrutinize models to see the gaps in the composition of structural layers, artifacts types and dependencies. The introduction of advance modelling techniques to automate EA practices also requires the need to test and validate those models. Further evidence shows that early frameworks such as the popular Zachman Framework (ZF), Generalized Enterprise Reference Architecture and Methodology (GERAM), Federal EA Framework (FEAF), The Open Group Architecture Framework (TOGAF) did not create extendable models that support validation (Fischer et al., 2010; Salmans et al., 2010; Urbaczewski et al., 2006). Consequently, attempts to introduce them at a later stage through improved versions and sometimes outright extension such as TOGAF with ArchiMate leave inconsistencies, omissions and gaps between the various layers of abstraction. Most practitioners tend to concede that it is not possible for a single Enterprise Architecture framework to completely present the enterprise in its entirety as comprising of business objectives, business processes, roles, organizational structures, organizational behaviors, information, software applications, computer systems and the relationships between these various entities (Chen et al., 2008). Though efforts still continue to be made towards standardization (TOGAF, 2012, OMG, 2012), many frameworks are specific in scope and purpose and apply to specific domains, generally weighted towards planning and business process analysis without commensurate emphasis on validation and change management.

3 Behavior-Driven Modelling Approach

Primarily, behavior-driven modelling focuses on behavioral specification of the abstracts of a model from a specific viewpoint of a stakeholder. Essentially, it specifies that for each abstract of a model under test, the following activities can be achieved (Essien, et al, 2013):

- Definition of a test set for the abstract;
- Implement the test set and; finally
- Verification that the implementation of the tested abstraction yields set goals.

This definition is specific as it allows validation to be carried out in terms of high-level goals specifications leaving out low-level requirement details and unrelated model components exertions. By this approach, validating a model allows specific structural test for appropriateness of model taxonomy to be made more specifically rather than depend on inferences from generic maturity matrices. The advantage is that it allows a model to be used for its ability to create value, enhance compliance visibility and change management rather than another metaphorical documentation requirement. Validating a model by concept of Behavior-driven modelling ensures that tests of any abstraction are specified in terms of the desired behavior of the abstracted component. The “desired behavior” in this case consists of the requirements set by the business, derived from the enterprise motivation concerns and restricted by its constraints. The validation scenarios are phrased declaratively rather than imperatively in a ubiquitous language with no reference to elements of the user interface through which the interactions take place (Mabey 2008). Following this approach, reference is made to how the desired behavior of the model abstract should be specified. To actualize this, the approach propfers the use of a semi-formal format for behavioral specification borrowed from problem domain specifications of the object-oriented analysis and design to develop a conceptual model that can be used to complete tasks that realize specific goal. This method recommends specification of behavior in terms of concerns and specifies the business value, explicitly written and annotated with constraints (Essien et al., 2013). The problem domain specifications adopts the following steps

**Title**

- The problem domain should have a clear, explicit title.

**Narrative**

A short, introductory section that specifies:

- The primary stakeholder of the problem domain which can be the actor, department or business process that benefits from the model abstract validation.
- Goals that are derived through the business behavior
- The components that are to be validated
- The relationship between the components, business behavior and the goal.

**Validation criteria or scenarios**

A description of validation test carried out on a specific case. Such a scenario would have the following structure:

- It starts by specifying the initial condition that is assumed to be true at the beginning of the scenario. This may consist of a single clause, or several.
- It then states which event triggers the start of the scenario.
- It states the input parsed to validate the component attributes, how the input is accepted, and the logic that characterizes the component and response channels.

**Acceptance criteria or scenarios**

- Finally, it states the expected outcome, in one or more clauses.

4 Specifications for Component Validation and Data Requirement

Model Validation by behavior specification validates scenarios of instance model from a logical model incremental-ly, across different views and layers of EA by testing component attributes against goals in the motivation Extension. This method though deploys the same principles of Behavior Driven Development (BDD) (North, 2006), differs as it focuses on behavioral specification of the EA artifacts rather than objectives. Three steps are iterated:

- Specification of model validation rules;
- Validation of the rule on the model instance;
- Validation of result with motivational goal;
Validation of artifact is based on the desired behavior with attributes set for the related motivational goal. The approach uses structure and behavior to establish patterns for traceability thus ensuring that the right design decisions are taken at the modeling stages. Not only can the method improve the quality and design of the framework, it also simplifies the alignment process. The validation scenarios describe the behavior and attributes of the component to be validated in order to realize set motivation goal thus ensuring better conformity with set goals. With the approach two aspects of model validation consisting of active and passive levels are identified. The Active Level of Model Validation is focused on the functionality of the artefact as defined by the scope of specified validation metrics and component under test. It identifies of business behaviour that the EA model artefact is expected to exhibit. The input (test) data is conceptualized based on the behaviour specified and goals to be achieved. On execution of a test scenario, the actual result based on the behaviour specified is determined and compared with expected goal. The Passive level of valuation examines the collaboration between models and interactions with different parts of the Business Architecture. This includes:

- The interface of business artefacts with motivational goals from view abstractions.
- The interface of business artefacts with constraints to enforce cohesiveness.
- Interactions on a view from views within the same viewpoints.
- Collaboration from views within the same viewpoint to achieve same goals.

Validation Data for a model is data specifically identified for use in testing the model. In this context, validation data are grouped as attributes of the model components and specifications from the business behavior; consisting of stakeholders concerns, test scenarios and constraints and expected outcomes. The component’s attributes may also consist of meta-data definitions for the object and relationship types with other artefacts. The test scenarios contain data segregation from constraints and goals in relationship with some measurement metrics. Figure 1 illustrates this classification.

**Figure 1: Classification of Validation Data**

Test scenarios specify procedural data used in a confirmatory way to verify that a given set of constraints to a given business behavior for given goal produces expected result. Test data is produced in a focused or systematic way adopting the VPEC-T concepts.

5 Modelling Languages and Validation

Modelling Language (ML) is a high level abstraction language, aimed at representing structures, characteristics and properties at early stage of design (Chen et al., 2008). Over the decade, there has been proliferation of ML as means of presenting visual images of design concepts. The Unified Modeling Language (UML), one of such adopted by Object Management Group (OMG) is a standardized, general-purpose modeling language in the field of software engineering and includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems (OMG, 2013). Though it combines techniques from data modeling (entity relationship diagrams), business modeling (work flows) and object modeling, it lacks the versatility that can visualize the entire enterprise as defined by “IEEE, Lankhorst, DODAF”. UML is focused on definition of system structure and behavior and has no built-in testing constructs (Samuelsson, 2003). The UML Test profile currently proposed is at a much lower level of abstract based on Testing and Test Control Notation Version 3 (TTCN3) and JUNIT than required in business behavior validation at EA higher abstraction. The Zachman framework (ZF) as an EA approach is a normalized six by six classification schema for organizing descriptive representations of an enterprise (Zachman, 2008). The rows represent distinct stakeholder perspectives of an enterprise, while the columns describe different areas of interest within those perspectives. The Zachman framework is simply a framework rather than a process, a method, a notation or a tool. Consequently the framework is rigid as rows and columns cannot be added or omitted to allow validation or testing. The Zachman framework is useful more in an enterprise as a general assessment tool for organizing a complete and holistic set of existing architecture descriptions and artifact sets; and to identify gaps in information and focus development efforts to fill the gaps. In ZF, there is no correct modelling tool for any particular cell. Any modelling tool may be used to depict the structural components of the cell e.g., UML diagrams, analytic equations, functional flow block diagrams, block diagrams of linear systems theory, transfer functions, state-space models, differential equations, object-oriented models, etc (Bahill, 2006). Each entity in the cell may use any representation for functions, processes, events, objects, data and interfaces. This makes it very difficult for components within ZF cells to share homogeneous annotations, semantics and relationship thus application of a standardized validation method on its framework is impracticable. The Extended Enterprise modelling Language (EEML) was developed as a comprehensive and generic framework for evaluating SEMiotic QUALity (SEQUAL) framework, (Krogstie, 2010). SEQUAL framework is systems modelling top-down reference model for evaluating the quality of models. It distinguishes between goals and means by separating the expected result from procedures needed to achievement it, through a process based on linguistic and semiotic on real world view with participation of the stakeholders. The core of the framework includes the discussion on syntax, semantics, and pragmatics parallel to the use of terms in the semiotic (Krogstie, 2010). EEML is under further developed in the EU projects Unified Enterprise ML (UEML) to validate and disseminate a set of core language for its support as a basis for interoperability within a smart organization (OMG, 2013). Consequently, it is unsuitable for EA valida-
tion as it is also domain specific. The representation of business rules is dependent on prevalent use and implementation. Maintenance and knowledge enhancement is a key requirement to its usability. Another popular means of validating the EA is the use of maturity matrix commonly referred to as Dynamic Architecture Maturity Matrix (DYAMM) (Noran, 2003). DyAMM is used as an instrument to assess the level of Enterprise Architecture Maturity (EAM) in organizations and covers many key areas that represent a different dimension within EAM. The DyAMM assessment method makes it possible to assess organizations on an overall maturity level as well as a specific level. The information for assessments is gathered through a survey questions that relate to one of the identified key areas (Coleman, 2006). However, this method which is purely qualitative has limitations as lack of a comprehensive approach to data gathering can affect judgement. The researcher’s presence during data gathering, which is often unavoidable, can affect the subjects’ responses. Issues of ethics, anonymity and confidentiality can present problems when presenting findings. Among other concerns, the questionnaire items are sometimes not well understood by the respondents and the CIO bias undoubtedly may influence the outcome. The Open Group Architecture Framework (TOGAF) is one of the most popular frameworks in EA. To provide a uniform representation for diagrams that describe enterprise architectures, the ArchiMate EAML is developed to support TOGAF Architecture Development Method (ADM) and to offer an integrated architectural approach that describes and visualizes the different architecture domains, their underlying relations and dependencies (TOGAF, 2012). However, assessment methodology is not integrated with ArchiMate Core or its extension. Maturity assessment discussed earlier is also deployed in TOGAF ArchiMate to identify the level of compliance between business vision and business capabilities. However, we adopt TOGAF and ArchiMate for the implementation of EA validation on the basis that ArchiMate actually sets the platform for achieving this by offering formal descriptions of components that support reasoning about the structural and behavioral properties of the organization. ArchiMate provides a graphical language for the representation of EA models and enables the introduction of annotations and semantics for validation.

6 Motivation Driven Validation Approach

The Motivation Driven Validation Approach (MDVA) validates viewpoints of a model iteratively, across the three aspects (Information, Business and Structure) of the ArchiMate business layer by testing attributes of the model elements against goals in the motivation Extension. This method though deploys the same principles of Behavior Driven Development (BDD), differs as it focuses on behavioral specification of the EA artifacts rather than requirements specifications. Three steps are iterated:
- Specification of model validation rules;
- Validation of the rule on the model instance;
- Validation of result with motivational goal;

Validation of artifact is based on the desired behavior with attributes set for the related motivational goal. The MDVA also establishes structural and behavioral patterns to ensure traceability thus ensuring that the right design decisions are taken at the modeling stages. Not only does the MDVA improve the quality and design of the framework through goals to component association, it also simplifies the traceability process. The validation scenarios for MDVA describe the behavior and attributes of the component to be validated in order to realize set motivation goal. MDVA ensures better conformance to user goals and provides the means for model traceability required for artifact validation.

6.1 MDVA Methodology

The MDVA consist of both the behavioral and the structural attributes of the EA components. Physical models of business behavior are created as derivative instances with different stakeholder perspectives for validation. Unlike BDD, test basis created are not based on the business behavior itself but on the attributes of the artifacts that constitute the model instance at a high level of abstraction.

6.2 MDVA Design

The MDVA is conceptualized from the ArchiMate Motivation Extension by deploying motivational element across the business and application layers. The methodology iterates correlations of motivational elements over the taxonomy to establish extent and coverage of the business behavior defined. Through the process, gaps and overlapping functionalities are identified allowing the model to be validated.

![Figure 2: An Overview of Validation Derivatives](image)

Figure 2 shows the artifacts that constitute the MDVA concept proposed in this paper. Validation themes are defined by set of metrics which specify what components are to be tested. By iterative refinement of business behavior on the Business Layer, components are modelled into views to aggregate viewpoints for a particular test requirement. Constraints are then applied on the components to derive test attributes for the logical model. These components form the test basis for the logical model design. The diagram depicts realization of physical model instances from the conceptual and logical models. These are validation from different views with test scenarios specified from constraints. Goals are part of the motivation extension of ArchiMate and ensure alignment and integration with the core EA. Through an iterative process, these models are revalidated through each test attribute of the artifact, generating traceability from specific view for each stakeholder.

7.0 The Case Study

The case study, grounded on Student Internship Programme (SIP) (SIP, 2012) at an academic institution is
used to illustrate the MDVA. A system is required with the aim to implement a program that offer student placement. The objective is to automate the process of matching students with employers and internships, allowing students to manage their CV, search for internship listings, request and apply for internship and store their feedback once the internship has taken place. The system allows employers to manage internship listings, track progress on internship listing and provide feedback on student internships once they have taken place. Administrators create users, search and match student CV with opportunities, forward student CV to employers, track student visits and generate reports on system usage. For this paper, a motivational model is required that can validate models created for student’s viewpoint.

7.1 ArchiMate Modeling
The ArchiMate language provides a means to handle modeling complexities of modern information-intensive enterprises. For our modelling concepts ArchiMate enterprise modeling language is used with the objective to extend the motivation attributes with tests specifications using the Model Driven Approach. Figure 3 shows the complete goals and aggregation refined for all the viewpoints in the SIP based on the requirements specified in the case study. Here the views of Student, Employer and Career Office are integrated to establish congruency and to ensure that there are no gaps. The student needs to be able to search for internship program and provide feedback; the employer needs to be able to provide the available internship opportunities as well as feedback; Career Office automates the match process as well as generates reports. The overall goal of the enterprise is to be able to guarantee that “Student Start Internship”. Constraints are modeled into the design and Goals are realized through Requirements.

7.2 Constraints
Though some of the constraints can also apply to the career office as the career office match students with placement opportunities also, we focus on student’s view and present constraint associated with students only to create our validation scenario.

Figure 4: Realization of Constraints within Goals.

In the following subsections, the preliminary process for implementation of the MDVA is carried out. Models from the motivational Requirements establish constraints and associations with Business Role, Business Function, Business Process and Business objects at the Business Layer. For the purpose of this paper, only the student’s abstraction will be used with associated Goals.

7.3 Business Role
Business role is used in a structural organizational sense to relate with Business processes or Business functions. Business Role is modeled in the Business Layer though it can be extended with components from both the Application and Technology layers and are assigned primarily to one or more business processes or business functions. The model created in Figure 5 for the case study illustrates the assignment of Business Function to a Business Role. While the Business Function assesses Business Objects, it triggers Events which initiates relevant Business Processes. Application Service is invoked through assess relationship in collaboration with Application Components and Data Objects.
7.5 Business Process

Business Process describes a flow of activities in the model represented in Figure 5 and 6. The Business Processes T1, T2 and T3 trigger the Business Function element T4 (Figure 6) represented as Business Process in Figure 5 and provides access to the Application service. Figure 6 shows aggregation and composite relationship attributes of the passive Business Function access relationship with Business Process. In the case study, the Business Process represents a workflow consisting of smaller processes leading to a Business Function “Apply for Internship”.

8 MDVA Implementation Approach

Harnessing the techniques described in previous sections, MDVA is grounded on the establishment of concrete test basis, defined at business level scenarios and annotated with constraints from the motivation concepts to support comparison between obtained and expected results. Figure 7 shows the transformation of Figure 5 relative to Figure 6 to include constraints defined in Figure 4.

In implementation of MDVA, we adopt the first step which is the creation of the conceptual model from the Goals requirements as a perspective to be validated, Fig 2 and Fig 7. Then, the next step transforms the conceptual model into a logical model based on constraint integrated into the taxonomy, specifying artefacts of the model that are to be tested. During this transformation, the test basis is generated explicitly including relations mappings with a traceability of model defined. The third step defines test scenarios with constraints and creates test conditions for validation of the EA artefacts. This is shown in Table 1 where in the model in Figure 4, constraints associated with the Goal “Start Internship” are extrapolated and cross-referenced with the business object model in Figure 6 and the architecture model in Figure 7. The constraints CM1, CM2, CM3, CM4 and CM5 are validated through CM constraints paths in the model transformation defined in Figure 7 and associated with corresponding objects in the actual implementation model. The constraints are applied on the defined artefacts to identify the existence of the object as well as validate stated conditions. Some of the test conditions defined in the goal motivation construct are exemplified on the table using business readable domain specific mnemonics.

Table 1: Application of Constraints to Model Artifacts.

At a higher level of abstraction, this can also be expressed using a BDD notation such as Gherkins for each of the constraints; For Example, CM1: BA1< T1>, BA3< T8> is validated as:

**Given** that artifact T1 exist in BA1  
**And** artifact T8 exist in BA3  
**When** Constraint CM1 is parsed in T1 and T8  
**Then** the result shall be True

A simple traceability model to demonstrate this notation usability is shown in Figure 8.

![Traceability model](Figure 8: Traceability model for artefact validation)

The MDVA technique addresses the traceability problem by creating relationships between transformed models and artefacts as part of the conversion process, externalizing the relationships among the test-artefact models to allow for comparison with expected outcome.

**9 Conclusions**

MDVA presented in this paper is an approach that decomposes business processes and develops constructs for the models to allow validation. Modelling motivational goals involve the conceptualization of different aspects of the enterprise from different viewpoints and levels of abstraction during the life cycle of the architecture. This article includes such conceptualizations derived through modelling and descriptions of models of the business behavior; specifying concepts of intentions in terms of goals, constraints and requirements. The models offer description of integrated components and illustrate the relationships between the various artifacts that constitute the taxonomy, relating business vision, mission and strategy with information systems through modeling extensions of ArchMate. Enterprise Architecture and its management have continued to be a topic of ongoing and increasing interest to practitioners. Standardization of concepts (considering disparities in ZF), methodology (as consolidated by TOGAF) would facilitate stabilization and leverage with new innovations to extend EA with validation models, notations and semantics. New technological trends such as cloud computing and big data pose challenge to EA integration. Creation of more EA management roles within enterprise needs to be embraced to allow evolution and provide more information for further research.

**REFERENCES**


[22]. Sammuelsson, E., 2003, UML Testing Profile RFP, Telelogic, Rational, SofTEAM, FOKUS, Ericsson,


