A Framework for Creating Pattern Languages for Enterprise Architecture

Paula Kotzé[1], Motse Tsogang[2] and Alta van der Merwe[3]

[1] CSIR Meraka Institute, PO Box 395, Pretoria, 0001, South Africa; and School of ICT Nelson Mandela Metropolitan University, Port Elizabeth, South Africa
[2] School of Computing, University of South Africa, Pretoria, 0003, South Africa
[3] Department of Informatics, University of Pretoria, Private Bag X20, Hatfield, 0028, South Africa.

paula.kotze@meraka.org.za, mtsogang@gmail.com, alta@up.ac.za

Abstract. The use of patterns and pattern languages in enterprise architecture (EA) is a relatively novel concept. Although both the concepts of patterns and EA are over 30 years old, the notion of design patterns is hardly applied to EA. There is a lack of pattern collections specifically devoted to EA: only a small number of patterns and pattern collections specifically aimed at enterprise architecture can be found in the public domain. Furthermore no framework or method exist that would assist enterprise architects in creating patterns and pattern languages for EA. This paper aims to bridge this gap by proposing a pattern framework for enterprise architecture (PF4EA), which can guide the development of well-grounded patterns and pattern languages for the EA domain. The components of the frameworks are described as well as a method for its use.

Keywords: Enterprise architecture, design patterns, pattern languages, pattern collections.

1 Introduction

Patterns are an attempt to describe solutions to problems or practices in a specific context, and which are harvested from 'best practices' and working solutions [18]. A *design pattern* is an approach to abstracting and capturing the knowledge for reuse on what made a solution, or paradigm, successful in relation to the problems identified in a particular context [35]. A design pattern can be thus be seen as "a piece of literature that describes a design problem and a general solution for the problem in a particular context" [14:2]. Design patterns originated in the field of building architecture, when Christopher Alexander invented the idea of capturing design guidelines in the form of design patterns [2]. Although the basic design pattern concept spans domains, the purpose, presentation and level of abstraction vary according to the domain and even within the domain [22]. Patterns are usually grouped into a *pattern collection*, either into a pattern catalogue or pattern language [6, 32, 35, 54]. This paper primarily focuses on pattern languages. A pattern language "is a collection of patterns that build on each other to generate a system" [14:17]. A pattern on its own solves a disjoint design problem, while a pattern language builds a 'system'.

The idea expressed in a pattern should be general enough to be applied in to a variety of systems within its context, but still specific enough to give constructive guidance. Design patterns are therefore often put forward as a way to assist novices in mastering a new domain [5, 12]. Patterns could likewise thus be put forward as a way to assist novice enterprise architects (and provide support for experienced enterprise architects) in the task of doing enterprise architecture.

An enterprise is a socio-technical organization or entity that functions on a relatively continuous basis to achieve a common set of goals and objectives, and has a mission and vision that guides how it should operate at all times [29, 37, 47]. An understanding of an enterprise's components and how they are related to one another can be obtained from its underlying architecture. *Enterprise architecture* (EA) "is the continuous practice of describing the essential elements of a socio-technical organization, their relationships to each other and to the environment, in order to understand complexity and manage change" [19].

Patterns and pattern languages for EA is a fairly novel domain. Although both the concepts of patterns and EA are over 30 years old, the notion of design patterns is hardly applied to EA and there is a lack of pattern collections specifically devoted to EA. To assist in bridging this gap, the aim of this paper is to propose a framework that can be followed to guide the development of well-grounded pattern languages for the EA domain. Although the framework is EA specific, the arguments on which the framework is based are fairly generic and can equally be applied to other domains (i.e. by replacing the EA-prefixed steps with a generic <topic>-prefix).

Section 2 provides the theoretical background for the paper by introducing the concept of patterns and pattern languages in more detail. Section 3 presents the Pattern Framework for Enterprise Architecture (PF4EA), whilst section 4 describes the method for using PF4EA. Section 5 provides examples of the use of PF4EA, whilst section 6 concludes.

2 Background

2.1 Patterns and Pattern Languages

Patterns are harvested from best practices on what has worked well in the past for a particular problem in a particular context, and is an attempt towards a description of successful implementation of a solution for that problem in the specific context [2, 32]. A pattern context is the preconditions under which a pattern is applicable, or a description of the initial state, before the pattern is applied to its intended problem [46]. From a usage perspective, patterns provide the guidelines for the description of solutions to analysis, design and architecture related problems [14, 18, 26]. In a practical sense, each pattern describes a problem that occurs repeatedly in a particular context, and then describes the core solution underpinning the problem, in such a way that one can use the solution many times over, without ever having exactly the same end result [2].

For any pattern to be legitimate, it must adhere to several general pattern characteristics [6, 14, 17, 54]:

- A pattern is grounded in a domain by being associated to a context as well as other patterns, and has no meaning outside the design domain or the pattern language it forms part of.
- A pattern implies an artefact.
- A pattern bridges many levels of abstraction.
- A pattern is both functional and non-functional, and should include the reason(s) and rationale why the solution is recommended, and what trade-offs are involved when such a pattern is used.
- A pattern is both a process and a thing, relating the design process and structure of the end product.
- A pattern is validated by use and cannot be verified or validated from a purely theoretical framework, without its practical application in its relevant context.
- A pattern captures a big idea and is meant to focus on key problems within a context and implies maximum reusability (whenever the problem emerges again, the pattern gets reapplied).
- A pattern conforms to a particular template.
- A pattern should be part of a pattern language where different patterns work together to solve a recurring complex problem in a particular context.

The next two sections discuss the pattern templates and pattern collections in more detail.

2.2 Pattern Forms and Templates

All patterns in the same language should have the same format [2]. A pattern form or template is a structure describing the essential elements and format of a pattern. Pattern templates vary between and even within application domains. For example, templates for building architecture (e.g. the Alexandrian Form for building architecture [41]), would differ from those for software engineering (e.g. the Portland Pattern Form (PF) [16], the canonical / Coplien form [3], the compact form (CF) [50], the Gang of Four Form (GoFF) [25], the Beck Form (BF) [44], etc.). In the EA domain the Enterprise Architecture Management (EAM) Pattern Catalog [21] supports a lightweight, organization-specific approach to EA management based on best practices, and distinguishes between three types of patterns: methodology (EA management) patterns, viewpoint patterns and information patterns. The pattern form are similar to the Buschmann's [9] software engineering form and includes the following elements: name, short description, example, context, problem, solution, implementation, variants, known uses, consequences, 'see also' (reference to associated patterns) and credits.

2.3 Pattern Collections

Patterns are usually grouped into a *pattern collection*, either into a pattern catalogue or a pattern language [6, 32, 35, 54]. A catalogue is a list or a collection of items usually organized in alphabetical order [48], where the patterns do not necessarily have to be related. When several related patterns are combined to solve a recurring complex problem in a specified context, the grouping of associated patterns is referred to as a *pattern language* [2, 7, 14, 18, 20, 35]. A pattern language is a structured method of describing good design practices within a particular domain. A pattern language is characterized by noting the common problems in a field of interest, describing the most effective solutions for meeting some stated goal, helping the designer move from problem to problem in a logical way, and allowing for many different paths through the design process.

2.4 Searching and Creating Individual Patterns

Patterns are discovered and not invented. There are basically two ways in which pattern collections can be discovered or formed [24]: through crafting/creating new patterns and through searching/harvesting patterns from existing pattern libraries or through automated processes (e.g. [45]). Patterns are discovered through observation and discrimination [24]. Observation reveals the underlying pattern and discrimination allows for selecting beneficial patterns that would advantage the specific domain. To craft a pattern, the problem to be solved must identified and the forces in tension discovered and documented. This is followed by a resolution of the forces, where the practitioner observes what solutions have been fashioned by other practitioners, and what is the best practice solution matching the forces that lead to the problem. The discovered solution is expressed as a pattern of action, which substantiates the solution in a general.

2.5 Creating Pattern Languages

Although patterns and pattern language collections abound, literature on the actual process of creating pattern languages are sparse. Cunningham [15], for example, suggested a few steps to get a pattern language writer going:

- Pick a whole area of focus, not just one idea. The area must practical and linked to the task that needs to be completed.
- Make a list of all the little things you have learned through the years about the area or document someone's experience in solving a particular problem.
- Cast each item on your list as a solution, and include the reasons for doing so (i.e. record the forces that bear on a solution).
- Write each item as a pattern making use of a pattern form (template).
- Organize the patterns into sections. Write an introductory paragraph to each section listing the patterns by name. Study the higher level structure of the patterns and write linking paragraphs when associations exist.
- Write an introduction to the patterns language, including the forces addressed.

In another example, Meszaros and Doble [40] defined a pattern language for writing patterns consisting of: context-setting patterns, pattern structuring patterns, pattern naming and referencing patterns, patterns for making patterns understandable and pattern language structuring patterns. The latter sets out a few guidelines for creating pattern languages:

- Identifying a set of patterns as a pattern language and writing a summary to introduce the larger problem and the patterns which contribute to solving it.
- Describe the overall context.
- Use a running example throughout.
- Highlight common problems, i.e. the common threads found in more than one pattern, and how the patterns can be used together to do something useful.
- Use distinctive headings to convey structure.
- Provide a problem/solution summary to help the reader find the pattern(s) that solve their specific problems
- Provide a glossary.

2.6 Patterns and Pattern Languages for Enterprise Architecture

Using the TOGAF architecture development process [47] as an example (but with no claim to representing the entire EA domain as such), the scope of the enterprise architecture development process is said to involve architecture vision development, business architecture development, information systems architecture development, technology architecture development, opportunity and solutions, migration planning, implementation governance, as well as the architecture change management. Enterprise architecture patterns should therefore include 'organizational' patterns that involve the full scope of enterprise architecture concerns, including people, processes, technology and facilities.

There are only a small number of pattern collections specifically focused on aspects of the EA development process, or claiming to focus on EA. Two existing examples, with individual patterns that are closely related to enterprise architecture from a primarily architecture management perspective, include:

- The EAM Pattern Catalog [21, 43] focusing on EA management to complement existing EA frameworks to provide a holistic and generic view on the problem of EA management, and to provide additional detail and guidance needed to systematically establish EA management in a step-wise fashion within an enterprise.
- A pattern catalogue for multichannel management described by Lankhorst and Oude Luttighuis [38], which they consider as a constituent of EA, to assist organisations to manage and align the various information channels they use in communicating with their customers.

Although limited specific EA patterns can be found, individual patterns can be found in disjoint pattern collections for other domains, which could be used in various enterprise architecture domains (but not specifically identified as such), for example organizational architecture [13], business modelling patterns [52], workflow patterns [51], software development patterns [25], etc. We also analysed a representative set of

EA frameworks and none supports the concept of design patterns in any substantive way. Design patterns are, however, briefly mentioned in TOGAF V8 [46], FEAF [11], The Zachman Framework for Enterprise Architecture [55] and GERAM [30]. There is therefore a lack of recorded research and guidelines on developing patterns for EA, and specifically pattern languages. In the case of pattern languages this is not the case for only the EA domain, but also in general. This paper attempts to address this gap in research by proposing a pattern framework for the development of patterns and pattern languages for EA, but which could also be used as guide to pattern language development in other domains.

3 The Pattern Framework for Enterprise Architecture (PF4EA)

Following an intensive literature study on the aspects that influence the development and maintenance of EA (combined with practical experience in these aspects), as well as an in-depth study and experience with the practices of patterns and pattern languages over an extended period of time, the Pattern Framework for Enterprise Architecture (PF4EA) was developed. PF4EA integrates the fundamental aspects related to patterns and pattern languages, as well as their associated processes and procedures, with the fundamental aspects related to enterprise architecture and its associated processes and procedures. **Error! Reference source not found.** Fig. 1 presents PF4EA graphically.

The components of PF4EA are organized into five construct layers, each addressing a specific aspect related to *patterns and pattern languages* and/or *enterprise architecture*:

1. *Theoretical context*: The theoretical context and best practices of both patterns and pattern languages and enterprise architecture, providing the theoretical foundation for PF4EA.
2. *Context specific rules and properties*: Determining and specifying the specific best practices, rules and properties related to patterns and pattern languages, which will be used in the patterns and pattern language to be developed, the specific enterprise architecture aspects for which the patterns and pattern language is to be developed, and the specific enterprise architecture framework(s) that will be supported by the patterns and pattern language to be developed in PF4EA.
3. *Context specific pattern relationships*: Specifying the context specific pattern relationships that will apply to the pattern language under development, including the *generic pattern relationships,* the *EA specific pattern relationships* and the related *EA framework specific pattern relationships.*
4. *Pattern search / creation*: Searching/creating individual patterns to support the aspects identified in the pattern context specific rules and properties making use of the EA processes and methodologies and EA framework rules and properties.
5. *Pattern language creation*: Applying the context specific pattern relationships to the set of standalone patterns created to develop a pattern language based on coherent principles. The output is the target pattern language for the specific enter-

prise architecture aspect under consideration. Each construct layer has an associated action that describes the action of use applicable to the construct layer, namely contextualize, consider, conform, create, and connect, respectively. These actions are described in more detail in section 4.

As indicated in Fig. 1, PF4EA comprises of 11 different components, which present the framework with various functionalities:

1. *Patterns, pattern languages and best practices (Component 1):* This component represents the theoretical foundation and the best practices related to patterns and pattern languages in general. It represents the generic pattern concepts to be con-

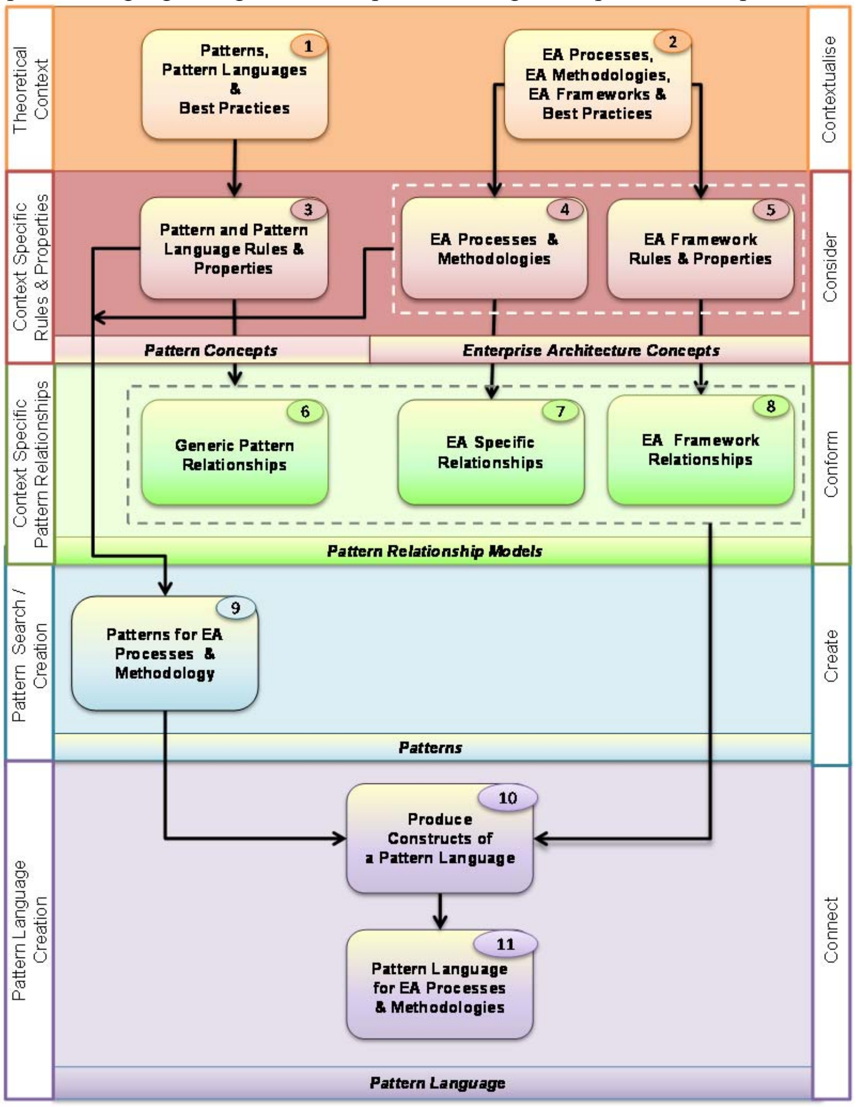

Theoretical Context
Contextualise
1
Patterns, Pattern Languages & Best Practices
2
EA Processes, EA Methodologies, EA Frameworks & Best Practices
Context Specific Rules & Properties
Consider
3
Pattern and Pattern Language Rules & Properties
4
EA Processes & Methodologies
5
EA Framework Rules & Properties
Pattern Concepts
Enterprise Architecture Concepts
Context Specific Pattern Relationships
Conform
6
Generic Pattern Relationships
7
EA Specific Relationships
8
EA Framework Relationships
Pattern Relationship Models
Pattern Search / Creation
Create
9
Patterns for EA Processes & Methodology
Patterns
Pattern Language Creation
Connect
10
Produce Constructs of a Pattern Language
11
Pattern Language for EA Processes & Methodologies
Pattern Language

Fig. 1. The Pattern Framework for Enterprise Architecture (PF4EA)

sidered for the composition of patterns for EA and the pattern languages for EA.

2. *EA Processes, methodologies, frameworks and best practices (Component 2):* This component represents the theoretical foundation and the best practices related to EA covering the generic concepts of EA processes, methodologies, frameworks and related best practices. It represents the generic EA concepts to be considered for the composition of patterns for EA and the pattern languages for EA. Both general EA concepts and EA framework detail are incorporated in this component, since the EA framework in use often 'prescribes' the process or methods to be followed in developing and EA or maintaining it.
3. *Pattern and pattern language rules and properties (Component 3):* This component provides the framework with selected context specific pattern rules and properties to govern the creation of patterns and their relationships in the pattern language to be developed. These rules and properties provide PF4EA with functionality to formalize the creation of patterns in a consistent manner through enforcement of specific rules, characteristic and properties of patterns and pattern relationships.
4. *EA processes and methodologies (Component 4):* This entails the detailed specification of the specific aspect of EA to be covered by the resulting pattern language. It specifies the conceptual foundation and specific methodologies related to the selected EA aspect to be considered.
5. *EA framework rules and properties (Component 5):* This component provides for all the rules and properties of the relevant EA framework(s) that will be supported by the resulting pattern language. EA frameworks provide the ground rules on the validity of connecting any two patterns in a pattern language.
6. *Generic pattern relationships (Component 6):* This component provides the valid generic pattern relationships by which one pattern can be associated to another in the resulting pattern language and what the nature of such a connection is. These pattern relationships are the essential aspects of producing pattern language constructs.
7. *EA specific pattern relationships (Component 7):* This component defines EA, or domain specific, pattern relationships. It specifies how a particular EA pattern may be linked to another through valid context specific pattern relationships.
8. *EA framework relationships (Component 8):* This component defines specific relationship semantics to support the selected EA framework(s). It thus provides for framework specific context relationships in the resulting the pattern language.
9. *Patterns for EA processes and methodologies (Component 9):* This component involves the creation of, or searching for, relevant individual patterns to support the EA concept under consideration.
10. *Pattern language constructs (Component 10):* This component involves identifying the relationships that exists between the individual patterns (identified in Component 9), using the generic pattern relationships (Component 6), the EA specific pattern relationships (Component 7), and the EA framework relationships (Component 8). This creates the individual pattern language pieces that when combined forms the pattern language for EA.

11. *Pattern language for EA processes and methodologies (Component 11):* This component integrates all of the patterns and the relationships that exists between them into a pattern language, and identifies any orphan patterns and gaps that may require the development of additional patterns or pattern relationships. In also includes a description of the overall context of the pattern language and provide a problem/solution summary and glossary.

4 Method to use PF4EA

For any framework to be complete, a method must be provided outlining the process to use the framework for its intended purpose. The use of PF4EA is categorized into five action stages, as indicated in Fig. 1:

1. *Contextualize*: Establishing the theoretical foundations and best practices supporting PF4EA.
2. *Consider:* Establishing and specifying the relevant pattern and EA aspects supporting, or to be supported by, the resulting pattern language.
3. *Conform:* Specifying how individual pattern components are allowed to relate to each other.
4. *Create:* Creating patterns for the EA concepts under consideration.
5. *Connect:* Connecting individual patterns into a pattern language.

Fig. 2 depicts the flow between these five actions and the steps through the related components when applying PF4EA to create a pattern language for the selected EA aspects. Each step is described briefly below.

1. *Contextualise*:
   - Step 1: Study fundamental patterns and pattern language theoretical concepts and best practices (if not familiar with this theoretical context already).
   - Step 2: Study the fundamental EA theoretical concepts (if not familiar with it already).
2. *Consider*:
   - Step 3: Use the knowledge obtained in Step 2 to determine the EA aspects for which a pattern language are to be created.
   - Step 4: Use the outcome of Step 3 to decide on the EA concept for which to create a pattern language. If the concept is not fully developed / specified generically, develop / refine the concept.
   - Step 5: Use the knowledge obtained in Steps 3 and 4 to decide on the EA framework that will be supported by the pattern language under development. If the EA framework rules and properties are not fully developed / specified generically, develop/refine the rules and properties.
   - Step 6: Use the knowledge obtained in Step 1 to decide on the general pattern rules and properties that must be adhered to by the pattern language to be developed, and which would be appropriate for the EA concepts identified in Steps 3 to 5.

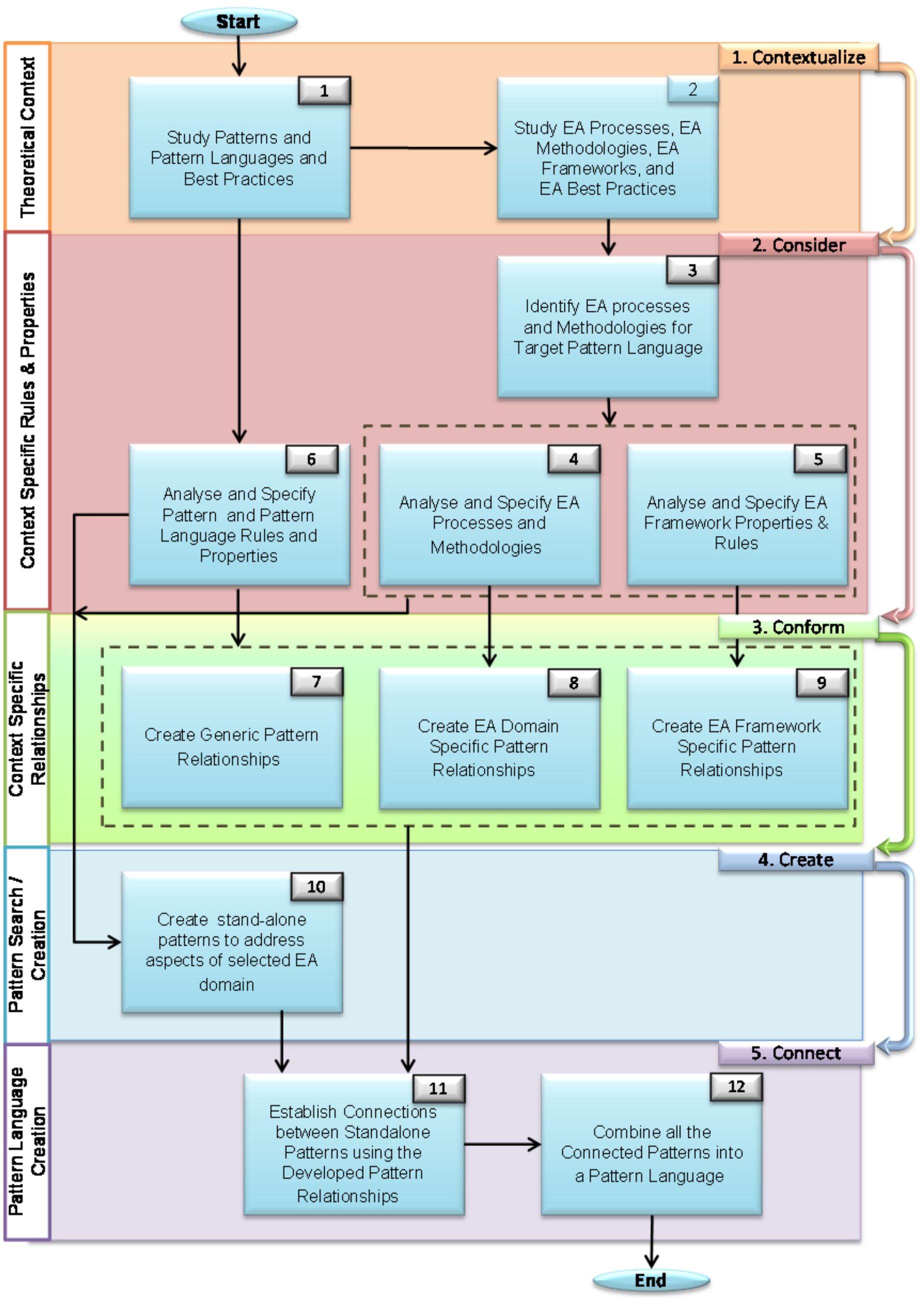

Start
Theoretical Context
1. Contextualize
1
Study Patterns and Pattern Languages and Best Practices
2
Study EA Processes, EA Methodologies, EA Frameworks, and EA Best Practices
Context Specific Rules & Properties
2. Consider
3
Identify EA processes and Methodologies for Target Pattern Language
6
Analyse and Specify Pattern and Pattern Language Rules and Properties
4
Analyse and Specify EA Processes and Methodologies
5
Analyse and Specify EA Framework Properties & Rules
Context Specific Relationships
3. Conform
7
Create Generic Pattern Relationships
8
Create EA Domain Specific Pattern Relationships
9
Create EA Framework Specific Pattern Relationships
Pattern Search / Creation
4. Create
10
Create stand-alone patterns to address aspects of selected EA domain
Pattern Language Creation
5. Connect
11
Establish Connections between Standalone Patterns using the Developed Pattern Relationships
12
Combine all the Connected Patterns into a Pattern Language
End

Fig. 2. Method to apply PF4EA

3. *Conform*:
   - Step 7: Use the outcome of Step 6 to specify the conditions for the generic pattern specific relationships that are to be used to connect individual patterns into a pattern language.
   - Step 8: Use the outcome of Step 4 and Step 6 to establish the EA specific pattern relationships matching the EA concept for which to create a pattern language.
   - Step 9: Use the outcome of Step 5 and Step 6 to establish the pattern relationships for the specific EA framework that will be supported.
4. *Create*:
   - Step 10: Search for existing patterns that support the EA concept identified in Step 4 and the EA framework identified in Step 5. If the patterns do not exist, develop/derive the patterns using the outcome of Steps 4, 5 and 6 using a pattern development method, such as the one described by Cunningham [15].
5. *Connect*:
   - Step 11: Using the outcome of Steps 7, 8 and 9 to establish relationships between the patterns identified or developed in Step 10.
   - Step 12: Use the outcome of Step 11 to combine all the patterns and graphically represent the resulting pattern language adhering to both pattern and EA fundamentals. If any orphan patterns exist, use steps 4 to 11 above to develop additional patterns enabling connections between all the patterns in the pattern language. Add a preamble to the resulting pattern language, describing the overall context of the pattern language, a problem/solution summary and a glossary.

5 Example – Towards a Pattern Language for Enterprise Architecture Development and Maintenance

To illustrate the components and use of PF4EA we present a number of examples. These examples, however, do not present the components of a complete pattern language, but are merely for illustrative purposes.

5.1 Contextualise

Steps 1 and 2
We have studied patterns and pattern languages in various domains in the past and are familiar with the basic concepts related to these aspects (see for example [33-35]). We are all experienced in EA and all have multiple of EA certifications and have published various papers on the topic (see for example [31, 39]). If this was not the case we would have had to study the basic concepts related to both the domains of EA and patterns and pattern languages in detail, prior to embarking on the pattern building exercise.

5.2 Consider

Step 3: Step 3 uses the knowledge obtained in Step 2 to determine the EA aspects for which a pattern language are to be created. Our aim was to develop a pattern language that could assist novice enterprise architects in the development and maintenance of enterprise architecture. We briefly introduce some of the concepts used in the remaining sections.

EA development focuses on establishing and specifying an understanding all of the socio-technical elements in an enterprise, including people, processes, business, organization and technology and how these elements interrelate [19, 53]. EA maintenance is a process of managing change to existing architecture models to accommodate changes that might have emerged due to change in process, technology, people and business [47].

Architectures have a cycle through which they evolve. According to Lankhorst [37], the architecture design process life cycle plays in important role in the evolution of any type of architecture. The architecture process consists of the steps that take an original idea through to the design and implementation phases, and eventually the management of architecture [4, 10, 27, 28, 37].

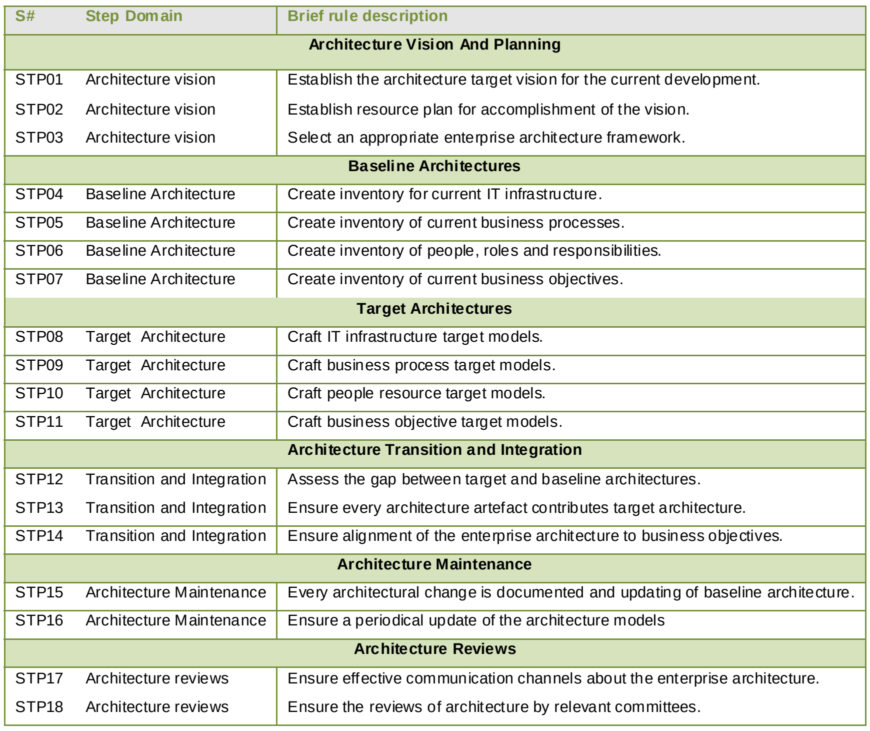
A *baseline architecture* is part of overall enterprise architecture and is an *as-is* overall architecture prior to entering a cycle of architecture review, redesign, development and maintenance [TOGAF, 2009]. A *target architecture* defines the *to-be-built* enterprise architecture and comprises of a complete description of the vision, scope, and partial high-level descriptions of the business's information systems models and design architectures reflecting the future view of aspects relating to business processes, data, applications, information systems and the technical infrastructure within an enterprise [11, 42, 49].

Zachman [55] defines EA as the total set of descriptive representations (models) relevant for describing an enterprise, that is, the descriptive representations required to create (a coherent, optimal) enterprise and to serve as a baseline for changing the enterprise once it is created. The total set of relevant descriptive representations would necessarily have to include all the intersections between the abstractions and perspectives. The Zachman Framework for Enterprise Architecture [56] do not formally define a process to use the Framework, but its use is implicated by the process to compile the descriptive representations mentioned.

Step 4: Step 4 uses the outcome of Step 3 to decide on the EA concept for which to create a pattern language. If the concept is not fully developed / specified generically, develop / refine the concept. An in depth study of EA concepts revealed that when referring to EA development and maintenance most authors refer to the TOGAF ADM [47], but that no generic set of steps for the development and maintenance of enterprise architecture exists. We therefore, as a first step, studied various publications and best practices to develop a set of generic steps for EA development and maintenance. An extract of these steps is presented in Fig. 3.

Step 5: Step 5 uses the knowledge obtained in Steps 3 and 4 to decide on the EA framework that will be supported by the pattern language under development. If the

EA framework rules and properties are not fully developed / specified generically, develop/refine the rules and properties. The Zachman Framework for Enterprise Architecture [56] was, due to its ontological nature and its ability to guide the development of applicable models, selected as an example to illustrate the concepts. The decision was made to use this framework for the first version of the pattern language. As a future endeavour a more comprehensive example using TOGAF is envisaged.



S#	Step Domain	Brief rule description
	Architecture Vision And Planning	
STP01	Architecture vision	Establish the architecture target vision for the current development.
STP02	Architecture vision	Establish resource plan for accomplishment of the vision.
STP03	Architecture vision	Select an appropriate enterprise architecture framework.
	Baseline Architectures	
STP04	Baseline Architecture	Create inventory for current IT infrastructure.
STP05	Baseline Architecture	Create inventory of current business processes.
STP06	Baseline Architecture	Create inventory of people, roles and responsibilities.
STP07	Baseline Architecture	Create inventory of current business objectives.
	Target Architectures	
STP08	Target Architecture	Craft IT infrastructure target models.
STP09	Target Architecture	Craft business process target models.
STP10	Target Architecture	Craft people resource target models.
STP11	Target Architecture	Craft business objective target models.
	Architecture Transition and Integration	
STP12	Transition and Integration	Assess the gap between target and baseline architectures.
STP13	Transition and Integration	Ensure every architecture artefact contributes target architecture.
STP14	Transition and Integration	Ensure alignment of the enterprise architecture to business objectives.
	Architecture Maintenance	
STP15	Architecture Maintenance	Every architectural change is documented and updating of baseline architecture.
STP16	Architecture Maintenance	Ensure a periodical update of the architecture models
	Architecture Reviews	
STP17	Architecture reviews	Ensure effective communication channels about the enterprise architecture.
STP18	Architecture reviews	Ensure the reviews of architecture by relevant committees.

Fig. 3. Extracts from EA development and maintenance process steps example

Step 6: Step 6 uses the knowledge obtained in Step 1 to decide on the general pattern rules and properties that must be adhered to by the pattern language to be developed and which would be appropriate for the EA concepts identified in Steps 3 to 5. Amongst other rules and properties, we decided on the use of the following pattern (expandable) template, derived from studying several other pattern templates:

— *Pattern Name*: A unique name to identify a pattern.
— *Problem*: The design problem which is addressed the creation of a pattern.
— *Context*: In which circumstances and domain is this pattern applicable?
— *Forces*: The various forces that impact the creation or existence of a pattern.
— *Solution*: Describe what needs to be done as a solution that resolves forces from strongest in this context in relation to addressing the recurring problem.

— *Related Patterns*: What enterprise architecture patterns are closely related to this one?
— *Rationale*: Is a description of why the solution is an appropriate one and not another.
— *Example*: An artefact (e.g. a graphical model, an algorithm, a formula, a structured rule (text), etc.), which illustrates how the pattern operates.

5.3 Conform

Step 7: Step 7 uses the outcome of Step 6 to specify the conditions for generic pattern specific relationships that are to be used to connect individual patterns into a pattern language. We specified a number of generic pattern relationships using a semi-formal notation. These relationships include the *is made of* relationship, *is equivalent of* relationship, *is alternative of* relationship and *is variant of* relationship [1, 8, 36].

For example, the *is equivalent of* relationship was specified using the following statements:

$\forall\ x{:}1..\mathrm{n},\ Pattern_x{:}\ PATTERN \bullet Pattern_x \Rightarrow Problem_x\ \wedge Solution_x\ \wedge Context_x$

(Given any pattern *Pattern*$_x$, there exist a problem *Problem*$_x$, being addressed by that pattern, and a solution *Solution*$_x$, produced by that pattern, and a context *Context*$_x$, in which such a pattern is applicable. The set PATTERN to represents the set of all valid patterns.)

$\forall\ i,j{:}1..\mathrm{n},\ Pattern_i,\ Pattern_j : PATTERN$
if $((Problem_i \equiv Problem_j) \wedge (Solution_i \equiv Solution_j) \wedge (Context_x \equiv Context_x))$
then $Pattern_i \equiv Pattern_j$
$\Rightarrow Pattern_i = \text{is-equivalent-of } Pattern_j$

(If the problems associated with *Pattern*$_i$ and *Pattern*$_j$ are equivalent, their solutions are and their contexts are also equivalent, then *Pattern*$_i$ and *Pattern*$_j$ are equivalent and said to be equivalent patterns of each other.)

Step 8: Step 8 uses the outcome of Step 4 and Step 6 to establish EA specific pattern relationships for the EA concept for which to create a pattern language.

We specified several EA specific relationships, one of which is the *is baseline2target of* pattern relationship. In this relationship, one pattern is used to produce a solution to a problem in the baseline architecture, whilst the second pattern is used to advance the baseline architecture into a target architecture solution.

$\forall\ i,j{:}1..\mathrm{n},\ Pattern_i,\ Pattern_j : PATTERN$
if $((Pattern_i \Rightarrow \text{Solution}_{\text{baselineArchitecture}}) \wedge (Patternj \Rightarrow \text{Solution}_{\text{targetArchitecture}})) \wedge$
$((\ Context_i\ \equiv Context_j) \wedge (Problem_i \equiv Problem_j)) \wedge$
$((Solution_i\ \text{«}\ Solution_j) \vee (\ Solution_i\, \text{»}\, Solution_i) \vee (\ Solution_i \equiv Solution_j\ ))$
then $Pattern_i = \text{is-baseline2target-of}(Pattern_j)$

if $Pattern_i\ = \text{is-baseline2target-of}\ (Pattern_j)$

then $Solution_i$ = is-baseline2target-of ($Solution_j$)
(« means the baseline architecture pattern solution remain unchanged whilst target architecture solution changes; » means the baseline architecture pattern solution changes into target with additional alterations, whilst ≡ means the baseline architecture pattern remains the same in the target architecture pattern solutions)

Step 9: Step 9 uses the outcome of Step 5 and Step 6 to establish pattern relationships for the specific EA framework that will be supported. We specified relationships for all the framework rules of The Zachman Framework for Enterprise Architecture [56]. These relationships include, amongst others: *diagonal*, *non-diagonal*, *is transformation of, is identification of, is definition of, is representation of*, *is specification of*, *is configuration of*.

For example, in The Zachman Framework for Enterprise Architecture, moving from one perspective to another in a vertical manner is referred to as transformation. The two patterns involved in an *is transformation of* type of relationship are associated with two adjacent perspectives ('rows' and abstractions (columns) in The Zachman Framework for Enterprise Architecture.

$\forall$ i,j:1..n, $Pattern_i$, $Pattern_j$: $PATTERN$
if (($Problem_i \neq Problem_j$) $\wedge$ ($Abstraction_i \equiv Abstraction_j$)) $\wedge$
  (($Perspective_i \neq Perspective_j$) $\wedge$ ($Solutioni \neq Solutionj$)) $\wedge$
  (($Perspective_i \wedge Perspective_j$) = adjacent)
then $Pattern_i$ = is-transformation-of($Pattern_j$)

if $Pattern_i$ = is-transformation-of ($Pattern_j$)
then : $Solution_i$ = is-transformation-of ($Solution_j$)

Although this approach is appropriate to The Zachman Framework for Enterprise Architecture, the approach of specifying the pattern relationships will have to be adapted for other frameworks, according to the rules of such frameworks.

5.4 Create:

Step 10: Step 10 searches for existing patterns that support the EA concept identified in Step 4 and the EA framework identified in Step 5. If the patterns do not exist, develop/derive the patterns using the outcome of Steps 4, 5 and 6. We created a (incomplete) set of patterns for the set of EA development and maintenance steps in Fig. 3. The set of patterns is indicated in Fig. 4. Fig. 5 illustrates an example of such a pattern.

EAP {Pattern language for enterprise architecture}	
EAP1=Architecture Vision Statement	EAP15=Target Enterprise Data
EAP2=Expert Resource Acquisition	EAP16=Target Information Systems Architecture
EAP3=Enterprise Architecture Framework Selection	EAP17=Target Technology Architecture
EAP4=Baseline Business Objectives Inventory	EAP18=Architecture Gap Examination
EAP5=Baseline Business Process Inventory	EAP19=Architecture Solution
EAP6=Baseline Enterprise Information Inventory	EAP20=Integration Implementation
EAP7=Baseline Human Capital	EAP21=Post Integration Architecture Examination
EAP8=Baseline Enterprise Data	EAP22=Architecture Change Management
EAP9=Baseline Information Systems Inventory	EAP23=Architecture Periodic Maintenance
EAP10=Baseline Technology Architecture Inventory	EAP24=Architecture Communication Glossary
EAP11=Target Business Objectives	EAP25=Architecture Communication Channel
EAP12=Target Business Process	EAP26=Architecture Committee Formation
EAP13=Target Enterprise Information	EAP27=Architecture Review Time Table
EAP14=Target Human Capital	

Fig. 4. Patterns for EA development and maintenance

Pattern EAP4

Pattern Name: *Enterprise Architecture Framework Selection*
Problem: How do you ensure that an appropriate enterprise architecture framework is selected for enterprise architecture development?
Context: you are doing enterprise architecture in which case you have to choose an appropriate and effective framework to use in the development of enterprise architecture.
Forces:

- The selection of a framework can be very challenging due to many existing and competing frameworks.
- The selection of a framework is dependent on its effectiveness in doing enterprise architecture and the experience of using such a framework.
- Select a good framework that is understood by all participants.
- Framework selection can be biased due to favouritism of one framework over another.

Solution:
Select an appropriate enterprise architecture framework that your expects have used and have experience in it. The framework will be used to create necessary enterprise architecture artefacts in accordance with desired architecture futuristic state.
Related patterns: In this pattern language, there is no specific related *enterprise architecture framework selection.*
Rationale:
It is crucial to have an inventory of where an enterprise is, in relation to available business objectives implementing any business objective to establish where the new objectives are going to fit in the baseline business objectives.

Fig. 5. EA Framework selection pattern

5.5 Connect:

Step 11: Step 11 uses the outcome of Steps 7, 8 and 9 to establish relationships between the patterns identified or developed in Step 10. Each individual pattern is compared to each of the other patterns to determine whether any relationship exists between the patterns. All the relationships are recorded. Although this may become a cumbersome process as the pattern language grows, the step is essential in establishing a valid pattern language. Further research would be required to streamline the process.

Step 12: Step 12 uses the outcome of Step 11 to combine all the patterns and graphically represent the resulting pattern language adhering to both pattern and EA fundamentals. The various perspective of The Zachman Framework for Enterprise Architecture [56] are used to guide the representation, e.g. the business architecture patterns are mapped to the business perspective, etc. The pattern relationships are then applied to all the patterns across all the perspectives to create meaningful associations between patterns mapped on the same perspective, and those with valid relationships in adjacent perspectives.

The process of connecting patterns to form the language involves the application of context pattern relationships, which associate one pattern to another via the type of relationship they share. If any orphan patterns exist, use steps 4 to 11 above to develop additional patterns enabling connections between all the patterns in the pattern language. Fig. 6 provides an example of how such a graphical representation of a pattern language for EA could be presented. The colour and shape of the connecting lines represent the various types of relationships that exist (e.g. ═ represents *is transformation of*). In addition to this representation a preamble to the resulting pattern language must be compiled (not shown here), describing the overall context of the pattern language, and provide problem/solution summary and glossary.

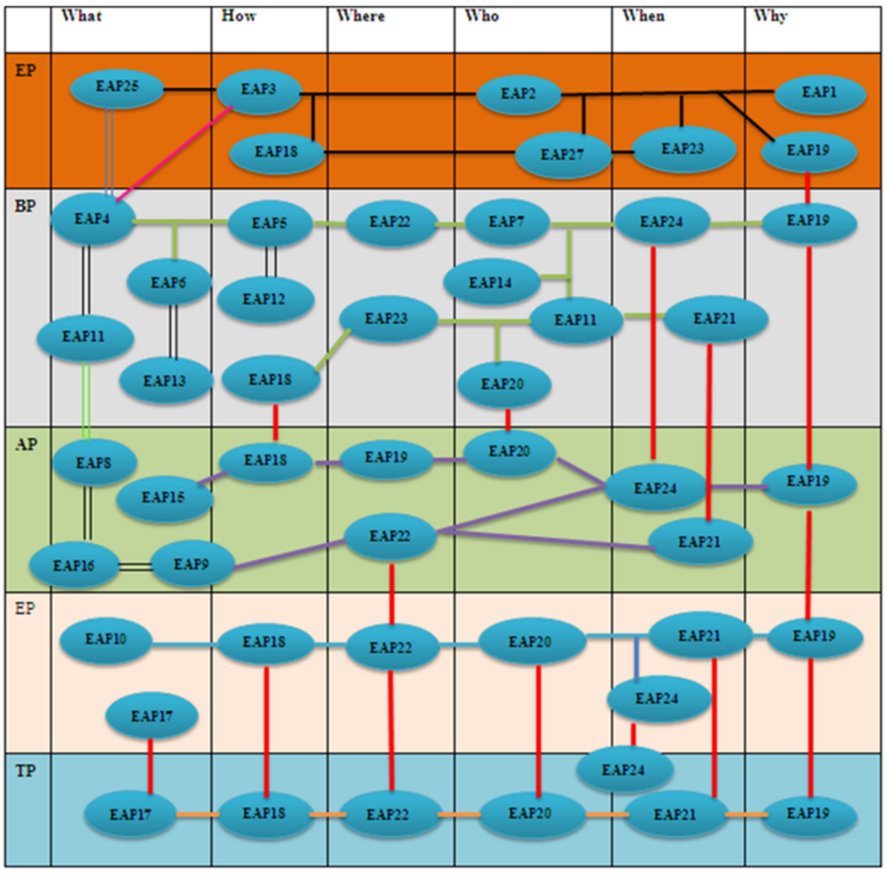


Fig. 6. Example of mapping a set of patterns for EA into a pattern language and graphically presenting the language using the abstractions and perspective of the Zachman Framework for Enterprise Architecture as canvas.

6 Conclusion

The rapid growth of the field of information and communications technology imposes change as the only constant faced by most enterprises. The biggest challenge that enterprises are facing currently is how to keep track of their internal and environmental changes as and when such occur. When an enterprise explicitly keep track of changes in its internal components and how these components interrelate to one another, as well as environmental change, it is said to have done its EA explicitly [23].

There are many approaches towards implementation of EA to assist enterprises to overcome their challenges relating to managing change and complexity. However the existing approaches do not explicitly include patterns as an approach to EA development and maintenance. The use of patterns and pattern languages in enterprise architecture is a relatively novel concept and only a small number of patterns and pattern collection specifically aimed at enterprise architecture can be found in the public domain. In this paper we presented a pattern framework for enterprise architecture (PF4EA), making use of, and augmenting some of the existing approaches to developing patterns and pattern languages (e.g. the work of Cunningham [15] and Meszaros and Doble [40] on creating patterns and pattern languages, respectively).

The purpose of PF4EA is to fill the gap in research for a baseline method that can be used in the development of patterns and pattern languages for EA. Using an example we illustrated how the use of the proposed framework can assist as a tool to understand the process of development and maintenance of patterns and pattern languages supporting the EA process. In future research a more comprehensive example, such as using TOGAF as selected framework, could complement this research and give more insight into the use of the framework.

References

1. Abel, A.: Design pattern relationships and classifications.: Computer and Information Science (2001)
2. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S.: A Pattern Language: Towns, Buildings, Construction Oxford University Press New York (1977)
3. Appleton, B.: Cononical Form. (2010)
4. Armour, F.J., Kaisler, S.H., Liu, S.Y.: Building an Enterprise Architecture Step by Step. IEEE Computer Society **1520-9202** (1999) pp. 31-38
5. Bergin, J.: A pattern language for initial course design. ACM SIGCSE Bulletin **31** (2001) 282 - 286
6. Bottomley, M.: A Pattern Language for Simple Embedded Systems.: Proceedings of PLOP 1999 - Pattern Languages of Programs'99. Hillside (1999)
7. Buschmann, F., Henney, K., Schamidt, D.: Past, Present, and Future Trends in Software Patterns. IEEE Software **24** (2007) 31 - 37
8. Buschmann, F., Henney, K., Schamidt, D.C.: Past, Present, and Future Trends in Software Patterns. IEEE Digital library. (2007)

9. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture : a system of patterns, Vol. 1. John Wiley & Sons, Chichester, New York (1996)
10. Callas, G.: Process Based Enterprise Architecture Building. IEEE Xplore Digital Library **1** (2006) pp. 239 - 244
11. CIO-Council: Federal Enterprise Architecture Framework Version 1.1. (1999)
12. Clancy, M.J., Linn, M.C.: Patterns and pedagogy. ACM SIGCSE Bulletin **31** (1999) 37 - 42
13. Coplien, J.: A generative development-process pattern language. Pattern Languages of Program Design. Addison-Wesley Publishing Co, New York (1994) 183 - 237
14. Coplien, J.: Software Patterns. SIGS Books & Multimedia, New York (1996)
15. Cunningham, W.: Tips for writing pattern languages. (1994)
16. Cunningham, W.: About the Portland Form. (2011)
17. Dearden, A., Finlay, J.: Pattern Languages in HCI: A Critical Review. Lawrence Erlbaum Associates, Inc. **21** (2006) 49–102
18. Devedzic, V.: Software Patterns. Handbook of Software Engineering and Knowledge Engineering. (2004)
19. EARF: Definition for enterprise architecture as defined by the Enterprise Architecture Research Forum. (2009)
20. Ellison, M.: A Pattern Language for Information Architecture. (2009)
21. Ernst, A.: Enterprise Architecture Management Patterns. PLoP '08: Proceedings of the 15th Conference on Pattern Languages of Programs ACM, New York (2008)
22. Faridul, I.: Investigating XML as Language for HCI Patterns Representation. Secondary Investigating XML as Language for HCI Patterns Representation. Vol. Master Thesis. Concordia University (2003)
23. Finkelstein, C.: Enterprise Architecture for Integration: Rapid Delivery Methods and Technologies Artech House (2006)
24. Fortino, A.: A Pattern Language for Innovation Management. Proceedings of PICMET 2008. IEEE, Cape Town, South Africa (2008) 415-419
25. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns, Elements of Reusable Object-oriented Software. Addison Wesley Professional, Boston (1995)
26. Griffiths, R.N., Pemberton, L.: Don't write guidelines write patterns! : (2005)
27. Gur, N.: Steps to create enterprise / system Architecture http://weblogs.asp.net/ngur/articles/194704.aspx (2004)
28. Harmon, P.: Developing enterprise architecture, white paper, business trends.: (2003)
29. Hoogervorst, J.A.P.: Enterprise Architecture : Enabling Integration, Agility and Change. (2003)
30. IFIP–IFAC Task Force on Architectures for Enterprise Integration: GERAM: Generalised Enterprise Reference Architecture and Methodology, Version 1.6.3. (1999)
31. Jacobs, D., Kotzé, P., van der Merwe, A., Gerber, A.: Enterprise Architecture for Small and Medium Enterprise Growth. In: Albani, A., Dietz, J.L.G., Verelst, J. (eds.): Advances in Enterprise Engineering V - First Enterprise Engineering Working Conference (EEWC 2011). Springer-Verlag, Berlin (2011) 61 -75
32. Kerievsky, J.: Refactoring To Patterns. Addison-Wesley, Boston (2005)

33. Kotzé, P., Renaud, K.: Do We Practise What We Preach in Formulating Our Design and Development Methods? In: van der Veer, G. (ed.): Lecture Notes in Computer Science, LNCS 4940. Springer, Berlin (2008) 566 - 585
34. Kotzé, P., Renaud, K., Koukouletsos, K., Khazaei, B., Dearden, A.: Patterns, anti-patterns and guidelines – effective aids to teaching HCI principles? In: Hvannberg, E.T., Read, J.C., Bannon, L., Kotzé, P., W, W. (eds.): Inventivity: Teaching theory, design and innovation in HCI. University of Limerick (2006) 115 - 120
35. Kotzé, P., Renaud, K., Van Biljon, J.: Don't do this – Pitfalls in using anti-patterns in teaching human–computer interaction principles. Computers & Education **50** (2008) 979–1008
36. Kumar, K., Prabhakar, T.V.: Design Decision Topology Model for Pattern Relationship Analysis. (2008)
37. Lankhorst, M. (ed.): Enterprise Architecture at Work: Modelling, Communication, and Analysis. Springer-Verlag, New York (2005)
38. Lankhorst, M., Oude Luttighuis, P.: Enterprise Architecture Patterns for Multichannel Management. Lecture Notes in Informatics (Software Engineering 2009) **P-150** (2009) 1-17
39. Mentz, J., Kotzé, P., van der Merwe, A.: A Comparison of Practitioner and Researcher Definitions of Enterprise Architecture using an Interpretation Method. In: Advances in Enterprise Information Systems. In: Moller, C., Chaudhry, S. (eds.), CRC Press/Balkema (2011)
40. Meszaros, G., Doble, J.: A pattern language for pattern writing. Pattern Languages of Program Design 3 Addison-Wesley Longman Publishing Co, Boston (1997) 529 - 574
41. Noyes, D.: Alexandrian Form. (2007)
42. Pavlak, A.: ENTERPRISE ARCHITECTURE : Lessons from Classical Architecture. (2006)
43. Sabine, Matthes, F., Schweda, C.M.: EAM Pattern Catalog Software Engineering for Business Information Systems, Faculty for Informatics,TU München (2010)
44. Silver, S.: Beck Form. (2007)
45. Spinellis, D., Raptis, K.: Component mining: A process and its pattern language. Information and Software Technology **42** (2000) 609-617
46. The Open Group: TOGAF 8.1.1 Online: Architecture Patterns. The Open Group (2006)
47. The Open Group: TOGAF Version 9. Van Haren Publishing, United States (2009)
48. Thompson, D.: The Concise Oxford Dictionary of current English. In: Flowler, H.W., Flowler, F.G. (eds.): The Concise Oxford Dictionary Clarendon Press, Oxford (1995)
49. TOGAF: Definitions. (2009)
50. Tremblay, B.: Compact Form. (2003)
51. van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow Patterns. (2003)
52. Veryard, R.: Component-Based Business Background Material, Business Patterns. (2000)
53. Wang, X., Zhao, Y.: An Enterprise Architecture Development Method in Chinese Manufacturing Industry. Ninth International Conference on Hybrid Intelligent Systems. IEEE Computer Society (2009) 226 - 230
54. Winn, T., Calder, P.: Is this a pattern? IEEE Software **19** (2002) 59-66

55. Zachman, J.A.: Z101 MasterClass: Framework Foundations (Presented as part of Zachman Certification training at CSIR's Meraka Intitute, 15 February 2010). Zachman International, , Pretoria (2010) 151
56. Zachman, J.A.: The Zachman Framework for Enterprise Architecture - The Enterprise Ontology Version 3.0. (2011)