Journal of Software Engineering and Applications, 2012, 5, 45-50
http://dx.doi.org/10.4236/jsea.2012.51008 Published Online January 2012 (http://www.SciRP.org/journal/jsea)

Pattern-Oriented Approach for Enterprise Architecture: TOGAF Framework

Mohamed Taleb[1], Omar Cherkaoui[2]

[1]École de Technologie Supérieure (ÉTS), Montreal, Canada; [2]University of Quebec at Montreal, Montreal, Canada.
Email: mohamed.taleb.1@ens.etsmtl.ca, cherkaoui.omar@uqam.ca

Received November 5th, 2011; revised December 8th, 2011; accepted December 20th, 2011

ABSTRACT

Design pattern suggests that developers must be able to reuse proven solutions emerging from the best design practices to solve common design problems while composing patterns to create reusable designs that can be mapped to different types of enterprise frameworks and architectures such as The Open Group Architecture Framework (TOGAF). Without this, business analysts, designers and developers are not properly applying design solutions or take full benefit of the power of patterns as reuse blocks, resulting in poor performance, poor scalability, and poor usability. Furthermore, these professionals may "reinvent the wheel" when attempting to implement the same design for different types of architectures of TOGAF framework. In this paper, we introduce different categories of design patterns as a vehicle for capturing and reusing good analyses, designs and implementation applied to TOGAF framework while detailing a motivating exemplar on how design patterns can be composed to create generic types of architectures of TOGAF framework. Then, we discuss why patterns are a suitable for developing and documenting various architectures including enterprise architectures as TOGAF.

Keywords: Design Patterns; Enterprise Architecture; TOGAF; Framework

1. Introduction

In recent years, many industrial firms have adopted architectures called enterprise architecture (EA). The Enterprise Architecture has matured from offering a lot of functionalities to like providing a clear representation of business processes and information systems, improving the IT governance, planning changes and optimizing resources.

Several definitions have been suggested by several authors. For example, The Institute for Enterprise Architecture Developments [1] "Enterprise Architecture is a complete expression of the enterprise; a master plan which acts as a collaboration force between aspects of business planning such as goals, visions, strategies and governance principles; aspects of business operations such as business terms, organization structures, processes and data; aspects of automation such as information systems and databases; and the enabling technological infrastructure of the business such as computers, operating systems and networks", Giachetti and MIT Center [2,3] "Enterprise Architecture is a rigorous description of the structure of an enterprise, which comprises enterprise components (business entities), the externally visible properties of those components, and the relationships (e.g. the behavior) between them. Enterprise Architecture describes the terminology, the composition of enterprise components, and their relationships with the external environment, and the guiding principles for the requirement (analysis), design, and evolution of an enterprise", the Enterprise Architecture Center of Excellence [4] "Enterprise Architecture explicitly describing an organization through a set of independent, non-redundant artifacts, defining how these artifacts interrelate with each other, and developing a set of prioritized, aligned initiatives and road maps to understand the organization, communicate this understanding to stakeholders, and move the organization forward to its desired state", and Ross *et al.* [5] "Enterprise Architecture is the organising logic for business processes and information technology (IT) infrastructure reflecting the integration and standardization requirements of the company's model".

All these definitions introduce the main architectural components (processes, systems, technologies, components and their relationships) and covers methods to represent them, including both functional and non-functional requirements, by means of a set of views.

Enterprise Architecture provides various benefits, such as 1) Well-established solutions to architectural problems of organizations; 2) Help in documenting architectural design and implementation decisions; and 3) Facilitation of collaboration and communication between users.

A number of industry standard approaches have been proposed for defining enterprise architecture, such as the Zachman Framework for Enterprise Architecture [6] and The Open Group Architecture Framework (TOGAF) [7].

In this technological context, we are borrowing, adapting and refining the so popular and powerful patterns-oriented development to enterprise architectures. The following are some of enterprise architectures challenges that we are addressing specifically while adapting the pattern-oriented approach to TOGAF framework. Furthermore, for a novice designer or a software engineer who is not familiar with this mosaic of guidelines, it is hard to remember all design guidelines, let alone using them effectively.

In this paper, we introduced different categories of design patterns as a vehicle for capturing and reusing good analyses, designs and implementation applied to TOGAF framework.

2. Background Work

Introduced by the architect Christopher Alexander in 1977 [8], design pattern can viewed as a building block that we compose to create a design. A single pattern describes a problem, which appears constantly in our environment, and thus described the hart of the solution to this problem, in a way such as one can reuse this solution for different platform, without ever doing it twice in same manner [8]. For the cross-platform application development, patterns are interesting for three reasons; see also [9] for a more general discussion on patterns benefits:

- They come from experiments on good know-how and were not created artificially;
- They are a means of documenting architectures (out of building or software, enterprise in general);
- They make it possible in the case of a cross-platform development in team to have a common vision.

Similar to the entire Enterprise Architecture community, the TOGAF community has been a forum for vigorous discussion on pattern languages for design, evaluation, and building a good architecture for the enterprises. The goals of the patterns is to share successful the design solutions among professionals and practitioners, and to provide a common ground for anyone involved in the design, development, enhanced usability testing, or the use of different systems. Several practitioners and designers have become interested in formulating various patterns of the same or different categories in the enterprise architecture destined to organizations.

The idea of using patterns in TOGAF Framework is not new. Different pattern collections have been published including patterns for layout design [10-12], for navigation in a large information architecture as well as for visualizing and presenting information. In our work, we investigate categories of Patterns as a solution for cross-platform Enterprise Architecture and in particular, to solve the following design challenges.

TOGAF [7] is an architecture framework that enables to design, evaluate, and build the right architecture for an organization. It is a mature Enterprise Architecture framework that is widely adopted by enterprises. TOGAF framework doesn't specify the architecture style—it is a generic framework TOGAF can be used in developing architecture. It consists of three main parts: The Enterprise Continuum, The TOGAF Resource Base and The TOGAF Architecture Development Method (ADM). ADM proposed a number of architectures shown and described below in **Figure 1**.

- Preliminary phase: This phase allows defining an Organization-Specific Architecture framework and the architecture principles. According the Dave Hamford [14], this phase is not a phase of architecture development;
- Phase A—Vision Architecture: This phase allows defining the scope of the foundation architecture effort, creating the vision architecture supporting requirements and constraints and obtaining approvals to proceed;
- Phase B—Business Architecture: This phase enables developing the detailed business architecture for analysing the gaps results;
- Phase C—Information System Architecture: This phase enables describing the Information Systems Architectures for an architecture project, including the development of Data and Application Architectures;
- Phase D—Technology Architecture: This phase enables developing a technology infrastructure that is used as a foundation for identifying all components that will support the development, implementation and deployment processes;
- Phase E—Opportunities and Solutions: This phase enables identifying opportunities and solutions and implementation constraints to deliver a more consistent architecture implementation;
- Phase F—Migration planning: This phase allows choosing and prioritizing all work packages, projects and to create, evolve and monitor the detailed implementation and migration plan providing necessary resources to enable the realization of the transition architectures;
- Phase G—Implementation Governance: This phase allows providing an architectural oversight of the implementation;
- Phase H—Architecture Change management: This phase allows establishing procedures for managing change to the new architecture;
- Phase Requirement Management: This phase allows

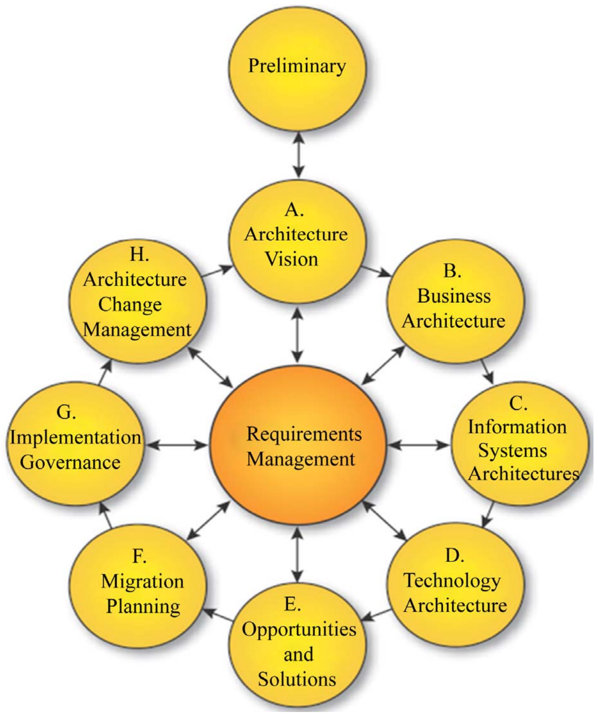
Preliminary
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

Figure 1. TOGAF framework [7].

managing architecture requirements throughout the Architecture Development Method (ADM), *i.e.*, defining a process whereby requirements for enterprise architecture are identified, stored, and fed into and out of the relevant ADM phases.

By combining different categories of patterns, the professionals and experts can utilize pattern relationships and combine them in order to produce an effective and coherence design solution by using fully service-oriented approach that TOGAF has adopted. As a result, patterns become a more effective vehicle that supports design reuse and building organizational capabilities.

3. The Proposed Patterns Taxonomy for TOGAF Framework

We propose at least ten categories of design patterns used to combine them to produce pattern-oriented enterprise architecture by applying the composition rules described in Section 4. Together, these patterns with their relationships provide an integrative solution to address the multi-faces of TOGAF Framework (**Figure 2**):

1) **Specification Patterns.** This category of patterns allows understanding and clarifying the adopted strategy context, goals, and business architecture principles to the stakeholders in order to coordinate, and integrate the specifications of different activities at different levels of the organization.

2) **Vision Patterns.** This category of patterns describes a clear and stimulating vision of architecture to develop for addressing its requirements and constraints, and to meet the defined goals and objectives. These patterns communicate share the information with stakeholders on the signification of aimed goals by the vision and emphasize its importance.

3) **Process Patterns.** This category of patterns coordinates the actions and operations that related together, in serial or in parallel manner, in order to reach a common objective. The actions are the activities executed par human. The operations are the activities executed and controlled automatically by a software system. When a process is composed only with operations, then we called it an automated process.

4) **Governance Patterns.** This category of patterns describes the manner that all architectures of TOGAF framework are well-governed and managed successfully by taking into account and addressing both potential risks and potential value of the enterprise architecture. These patterns provide and inform the proper functioning of these various architectures, and specially their deployment and interaction. Theses architectures are linked by sequential interdependencies form. Indeed, they exchange together to produce the desired outcomes. Information must propagate between the involved architectures during the execution to harmonize their efforts to obtain better governance.

5) **Migration Planning Patterns.** This category of patterns describes and explains the important strategies of migration plan that were proven with execution. This effective plan consists of four key steps such as definition of needs, design, implementation, and tests. In addition, these patterns have to address the details of overall aspects through these strategies to ensure the optimal quality of the migrated functionalities of systems by including the best practices in order to develop the detail of the target organizational architecture.

6) **Usability Patterns.** This category of patterns focuses on dealing with the relationships between internal software attributes and externally visible usability factors and how these patterns can lead to a methodological framework for improving the "Opportunities and Solutions" architecture, and how these patterns can support the integration of usability in the software design process. In addition, these patterns expose knowledge that has been gained from different projects by many experts over many years.

7) **Architecture Patterns.** This category of patterns describes and gives information about the type of technological infrastructure to develop. Indeed, these patterns will support and enable the different business services implementation and deployment by using Service-Oriented Architecture (SOA) components of TOGAF framework.

8) **Information Patterns.** This category of patterns describes different conceptual models and architectures

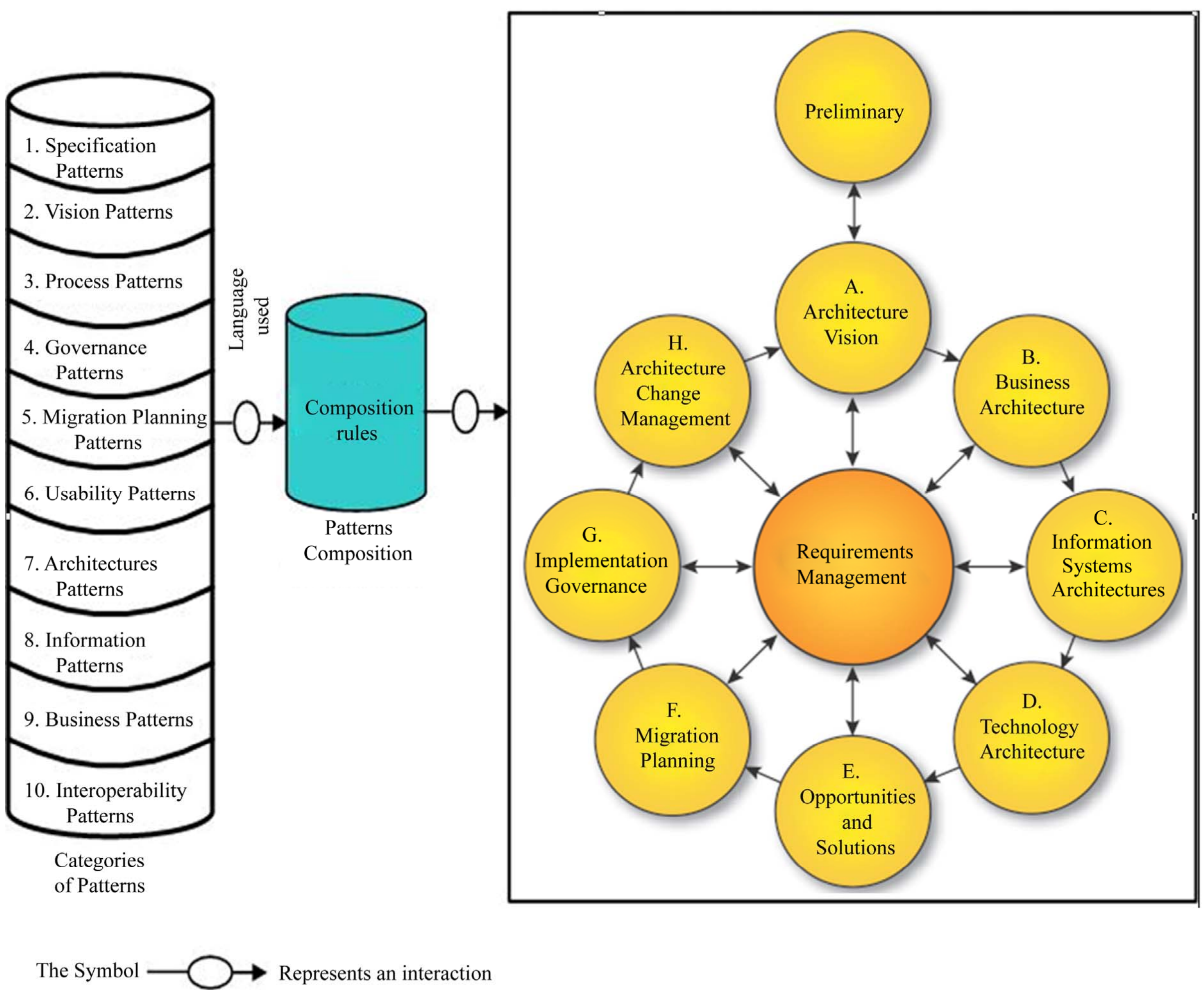

Figure 2. Pattern-oriented TOGAF framework.

for organizing the underlying content across multiple pages, servers and computers. Such patterns provide solutions to questions such as which information can be or should be presented on which device.

9) **Business Patterns.** This category of patterns describes a communication between the vision of organization with its business subjects with its objectives and its environment model such as actors, roles, and business service or functional or information or decomposition diagrams, business interaction, business footprint, product lifecycle diagram and all business processes involved.

10) **Interoperability Patterns.** This category of patterns is useful for decoupling the organization of these different categories of patterns as outlined in **Figure 2**, for the way information is presented to the user, and for the user who interacts with the information content. Patterns in this category generally describe the capability of different architectural programs to exchange data, via a common set of exchange formats considered as a service, to read and write under the same file formats, and to use the same protocols.

Communication and interoperability patterns are useful for facilitating the mapping of a design between architectures of TOGAF framework.

Gamma *et al.* [13] offer a large catalog of patterns for dealing with such problems. Examples of patterns applicable to interactive systems include: Adapter, Bridge, Builder, Decorator, Factory Method, Mediator, Memento, Prototype, Proxy, Singleton, State, Strategy, and Visitor.

4. Pattern Composition Rules

A creation of an Enterprise Architecture pattern oriented design exploits several relationships between patterns. Based on previous work [15], we identify five types of relationships.

1) **Similar** is a relationship, which applies to the same category of patterns. Two patterns (X, Y) are similar, or equivalent, if, and only if, X and Y can be replaced by each other in a certain composition. This means that X and Y are patterns of the same category and they provide

different solutions to the same problem in the same context. For example, the *Index Browsing* and *Menu Bar* patterns are similar. They both provide navigational support in the context of a medium-sized.

2) **Competitor** is a relationship that applies to two patterns of the same patterns category. Two patterns (X, Y) are competitors if X and Y cannot be used at the same time for designing the same artifact relationship that applies to two patterns of the same pattern category. Two patterns are competitors if, and only if, they are similar and interchangeable. For example, the Web patterns *Convenient Toolbar* and *Index Browsing* are competitors. The *Index Browsing* pattern can be used as a shortcut toolbar that allows a user to directly access a set of common services from any interactive system. The *Convenient Toolbar*, which provides the same solution, is generally considered more appropriate.

3) **Super-ordinate** is the basic relationship to compose several patterns of different categories. A pattern X is a super-ordinate of pattern Y, which means that pattern Y is used as a building block to create pattern X. An example is the *Home Page* pattern, which is generally composed of several other patterns.

4) **Subordinate.** If pattern X is super-ordinate of Y and Z then Y and Z are sub-ordinate of X. This relationship is important in the mapping process of pattern-oriented design from an architecture to another one. For example, the *Convenient Toolbar* pattern is a sub-ordinate of the *Home Page* pattern for either a PDA or desktop platform. Implementations of this pattern are different for different devices.

5) **Neighboring.** Two patterns (X, Y) are neighboring if X and Y belong to the same pattern category. For example, the sequential and hierarchical patterns are neighboring because they belong to the same category of patterns, and neighboring patterns may include the set of patterns for designing a specific page such as a home page

5. An Illustrative Example

This section describes the design patterns illustrating and clarifying the core ideas of the pattern-oriented approach and its practical relevance. This case study illustrates how patterns are used to formalize and design the requirements of various architectures constituent TOGAF framework.

In what follows, we have introduced some concrete examples of this mosaic of patterns that we have been using. These examples have shown also the need to combine several types of patterns to provide solutions to complex problems. The list of patterns is not exhaustive. There is no doubt that more patterns are still to be discovered, and that an endless number have yet to be invented.

Interoperability patterns are fundamental patterns to facilitate the communication between requirements management phase and other architectures of TOGAF framework. Example of patterns that can be considered to ensure the interoperability of architectures include Adapter, Bridge, Builder, Decorator, Facade, Factory Method, Mediator, Memento, Prototype, Proxy, Singleton, State, Strategy, Visitor [13].

The Adapter pattern is very common, not only to remote client/server programming, but to any situation in which there is one class and it is desirable to reuse that class, but where the system interface does not match the class interface. **Figure 3** illustrates how an adapter works. In this figure, the Client wants to invoke the method R*equest*() in the Target interface. Since the Adaptee class has no *Request*() method, it is the job of the Adapter to convert the request to an available matching method. Here, the Adapter converts the method *Request*() call into the Adaptee method *specificRequest*() call. The Adapter performs this conversion for each method that needs adapting. This is also known as *Wrappering*.

6. Discussion

The types of TOGAF architectures that are recommended for some the most popular patterns and which can be used to redesign of development systems for different architectures.

In this paper, we have introduced a pattern-oriented design method that essentially exploits different categories of patterns. This approach is a significant improvement over non-structured migration methods currently in use, for the following reasons:

- The method provides a standardized table of patterns, thereby reducing the redesign effort and ensuring consistency in redesign.
- The method helps designers in design choices associated with (1) the size of the source architecture and target architecture and (2) the amount of information to maintain in migrating from the source architecture to the target architecture.

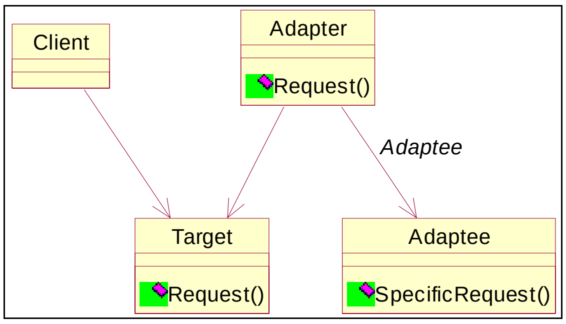

Figure 3. Adapter pattern.

- The method is simple enough to be used easily by novice designers, as compared to reengineering which currently requires a considerable degree of expertise and abstract reasoning ability.

Pattern-oriented approach offers the very useful ability of easily building multiple architecture-specific designs. However, the current state of the art in patterns and cross-architecture research is not yet mature enough to handle all the requirements of pattern-oriented design. More research must be addressed to define the multiple levels of abstraction of patterns and to create a clear, well-structured taxonomy of patterns. The simplified taxonomy presented in Section 3 is a starting point. Thus, within a pattern-oriented framework, the simplified "redesign and design" method proposed here is currently the most practical approach for migration of systems between architectures.

7. Conclusions

In this paper, we have identified and proposed ten categories of patterns, providing examples, for a pattern-oriented architecture for TOGAF framework to demonstrate when a pattern is applicable or required during the design process, how it can be reused and the underlying best practices to come up with reusable design solutions.

Our experiences highlighted also that in order to render the patterns understandable by novice designers and engineers who are unfamiliar with enterprise architecture, patterns should be presented to developers using a flexible structure to represent patterns, to make it easy for both the pattern authors, reviewers and users.

One of the major problems we can find is that mastering and applying several types of patterns require in-depth knowledge of both the problems and forces at play and most importantly must ultimately put forth battle-tested solutions. As such, it is inconceivable that pattern hierarchies will evolve strictly from theoretical considerations. Practical research and industry feedback are crucial in determining how successful a pattern-oriented design framework is at solving real-world problems. It is therefore essential to build an "academia-industry bridge" by establishing formal communication channels between industrial specialists in patterns, enterprise architecture design patterns such as TOGAF framework as well as pattern researchers. Such collaboration will lead, at to a common terminology which essential making the large diversity of patterns accessible to common TOGAF framework designers.

Future work will require the classification of each pattern and the illustration of each of them in UML class and sequence diagrams for each architecture of TOGAF framework. Next, some relationships will have to be defined between patterns so that they can be combined to create models based on the resulting patterns. Also, the design patterns need to be evaluated using different evaluation standards and methods and the formal descriptions of the proposed patterns using the formal language such as XML and its derivatives to increase the number of these formal descriptions which is also conducive to the future engineering application.

REFERENCES

[1] Institute for Enterprise Architecture Developments, 2011. http://www.enterprise-architecture.info/

[2] R. E. Giachetti, "Design of Enterprise Systems, Theory, Architecture, and Methods," CRC Press, Boca Raton, 2010.

[3] P. Weill, "Innovating with Information Systems: What Do the Most Agile Firms in the World Do?" *6th e-Business Conference*, Barcelona, 2007.

[4] Enterprise Architecture Center of Excellence, 2011. http://eacoe.org/index.shtml

[5] J. W. Ross, P. Weill and D. C. Robertson, "Enterprise Architecture as Strategy," Harvard Business Press, Boston, 2006.

[6] J. A. Zachman, "A Framework for Information Systems Architecture," *IBM Systems Journal*, Vol. 26, No. 3, 1987, pp. 276-292. doi:10.1147/sj.263.0276

[7] Open Group, 2008. http://pubs.opengroup.org/architecture/togaf9-doc/arch/index.html

[8] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiskdahl-King and S. Angel, "A Pattern Language," Oxford University Press, New York, 1977.

[9] F. Buschmann, "What is a Pattern?" *Object Expert*, Vol. 1, No. 3, 1996, pp. 17-18.

[10] J. Tidwell, "Common Ground: A Pattern Language for Human-Computer Interface Design," 1997. http://www.mit.edu/~jtidwell/common_ground.html

[11] T. Coram and J. Lee, "Experiences—A Pattern Language for User Interface Design," 1998. http://www.maplefish.com/todd/papers/experiences

[12] M. V. Welie, "The Amsterdam Collection of Patterns in User Interface Design," 1999. http://www.cs.vu.nl/~martijn/patterns/index.html

[13] E. Gamma, R. Helm, R. Johnson and J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software," Addison-Wesley, Reading, 1995.

[14] Dave Hornford, SOA/TOGAF Tutorial, 2010. https://www.opengroup.org/conference-live/uploads/40/22062/hornford.pdf

[15] F. J. Budinsky, M. A. Finnie, J. M. Vlissides and P. S. Yu, "Automatic Code Generation from Design Patterns," *IBM Systems Journal*, Vol. 35, No. 2, 1996, pp. 151-171. doi:10.1147/sj.352.0151