Relating Patterns and Reference Architectures

Eduardo Guerra, LAC/INPE, Brazil
Elisa Yumi Nakagawa, ICMC/USP, Brazil

Abstract: *Both patterns and reference architectures aim to describe solutions to be reused for the software systems development. Despite they have a common goal, they have a lot of differences and have also been investigated separately. The objective of this paper is to discuss the relationship between them, and how they can be complementary, respecting their respective peculiarities. We also discuss how patterns can support the creation of reference architectures and how reference architectures can be source for pattern mining.*

1. Introduction

Patterns are recurrent solutions that are used for problems on different software systems that share a specific context. On the field of software design and architecture, patterns present technical solutions on how to structure software components in order to achieve desired requirements. On the other hand, a reference architecture is a special type of software architecture that can be used as a basis for the construction, standardization, and evolution of software architectures of systems in a given domain. It defines elements, their roles and relationships needed to solve common problems in aimed domains.

Based on their description, it is possible to perceive that these two ways of documenting reusable solutions have a lot in common with each other. However, despite few isolated works that focus on specific domains [Striker et. al 2010; Guerra et. al 2013B; Fernandez et. al 2015], this relationship is not being explored. The goal of the paper is to explore the relationship between patterns and reference architectures, and present a discussion to work with both in a complementary way and, as a consequence, aiming at supporting more effectively the development of software systems.

The paper is organized as follows: Section 2 presents an overview of reference architecture concepts; Section 3 describes a process for the reference architecture creation; Section 4 highlights differences between patterns and reference architectures; Section 5 focuses on guidelines for using a pattern language as the basis of reference architectures; Section 6 presents some research perspectives in this topic; and, finally, Section 7 concludes the paper.

2. An Overview of Reference Architectures

A reference architecture refers to an architecture that encompasses the knowledge about how to design concrete architectures of systems of a given application domain; therefore, it must address the business rules, architectural styles (sometimes also defined as architectural patterns that can also address quality attributes in the reference architecture), best practices of software development (for instance, architectural decisions, domain constraints, legislation, and standards), and the software elements that support development of systems for that domain. All of this must be supported by a unified, unambiguous, and widely understood domain terminology [Nakagawa, 2015].

Reference architectures usually contain important knowledge about how to organize architectures of software systems of a given domain, presenting several possibilities of use. Diverse purposes have guided their adoption and use, such as the creation of concrete architectures to support the building of software systems, standardization of systems of a given domain, evolution of existing software systems, construction of new reference architectures, and support to the building of software product line.

Considering their relevance, different domains have already understood the need for encapsulating knowledge in reference architectures, with the aim at disseminating and reusing this knowledge and standardizing the systems as well. Good examples are AUTOSAR (AUTomotive Open System ARchitecture) [Autosar, 2015], for the automotive domain, and Continua [Continua, 2015] and UniversAAL [Universaal, 2015], for AAL (Ambient Assisted Living). In particular, these architectures have been developed by consortiums that involve major industrial players (such as manufacturers and suppliers) and researchers. Besides that, platform-oriented architectures, i.e., architectures not related to the specific application domain, but to a specific architectural style or technology, have also been proposed and widely used as reference architectures. Some good examples are OASIS [Oasis, 2008] and S3 [Arsanjani, 2007], for software systems based on SOA (Service-Oriented Architecture) [Papazoglou, 2008], and OSGi (Open Services Gateway initiative framework) [OSGI, 2015], a set of specifications and a layered architecture that define dynamic component systems for Java. However, it is observed that these architectures were created without using a systematic approach, what can be consuming extra effort, time, and even requiring rework.

In order to systematize the establishment of reference architectures, some initiatives can be found, such as that ones found in [Angelov, 2012; Bayer, 2004; Cloutier, 2010; Dobrica, 2008; Galster, 2011; Muller, 2008]. They are in general high-level guidelines, principles, recommendations or domain-specific processes. In this scenario, we proposed ProSA-RA [Nakagawa, 2014], a process for the building of reference architectures, focusing on how to design, represent, and evaluate such architectures. It has been widely used to create several architectures for different domains and scenarios (both industry and academia); however, we had not still explored the use and benefits that could be provided by patterns.

3. Reference Architecture Creation

Similarly to processes to construct software architectures, reference architectures should be built considering a set of systematized steps, as already discussed by the software architecture community. The goal of this section is to present one of such systematizations, i.e., a process that has supported the building of reference architectures. .

From our experience in building reference architectures, as well as in systematizing the steps to build them, we can summarize the whole building process as presented in Figure 1. To adequately apply this process, first of all, it is specially important to establish the scope of the reference architecture, i.e., the target application domain. This scope can be defined considering the set of systems that are intended to be produced based on this architecture. In short, four steps compose this process:

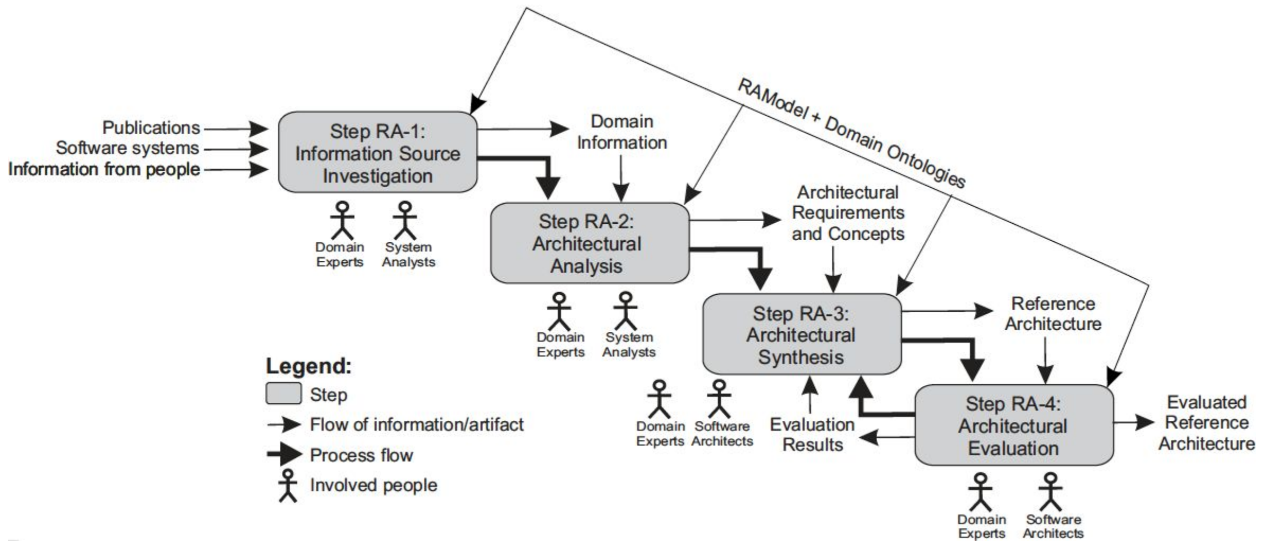
RAModel + Domain Ontologies
Publications
Software systems
Information from people
Step RA-1: Information Source Investigation
Domain Experts
System Analysts
Domain Information
Step RA-2: Architectural Analysis
Domain Experts
System Analysts
Architectural Requirements and Concepts
Step RA-3: Architectural Synthesis
Domain Experts
Software Architects
Reference Architecture
Evaluation Results
Step RA-4: Architectural Evaluation
Domain Experts
Software Architects
Evaluated Reference Architecture
Legend:
Step
Flow of information/artifact
Process flow
Involved people

Figure 1: Outline Structure of ProSA-RA [Nakagawa 2012A]

- Step RA-1: Information Source Investigation: In this step, the main information sources are selected. These sources must provide information about processes and activities that could be supported by software systems of the target domain. Since reference architectures should be basis of several software systems of a given domain, these sources must involve a more comprehensive knowledge about the domain if compared with information sources when developing the architecture of a specific system. In this perspective, ProSA-RA highlights as the most relevant sources: people, documents, systems of the domain, related reference models and reference architectures, and domain ontologies, if available;
- Step RA-2: Architectural Analysis: Based on the selected sources, three set of elements are identified. Firstly, the set of requirements of software systems of that domain is identified and, based on these requirements, the set of requirements of the reference architecture is then identified. These requirements include functional and non-functional requirements. After that, the set of concepts that must be considered in this reference

architecture is established. For this, three main tasks are performed: (i) identification of the system requirements; (ii) identification of the reference architecture requirements; and (iii) identification of the domain concepts;

- Step RA-3: Architectural Synthesis: In this step, the architectural description of the reference architecture is built following a model for reference architectures, such as RAModel [Nakagawa, 2012]. For instance, architectural styles, as well as a combination of these and other styles, probably identified in Step RA-1, must be considered. These styles are the basis on which concepts previously identified are organized. For instance, if a three-tier architecture is used as an architectural style of the reference architecture, the concepts related to the business rules should be organized into the application tier, since this tier usually contains these rules. Besides that, considering the effectiveness of using architectural views to represent concrete software architectures, they can be also adopted to describe reference architectures; and
- Step RA-4: Architectural Evaluation: In the context of this work, reference architecture evaluation refers to the task to check the architectural description of such architecture together with diverse stakeholders intending to detect defects in this description. For this, ProSA-RA uses a checklist-based inspection approach, called FERA (Framework for Evaluation of Reference Architectures). In short, FERA is composed by 93 multiple choice questions whose answers vary from ``fully satisfactory'' to ``totally unsatisfactory'', and fields to add comments. It also considers to be answered by a range of stakeholders (namely architects, domain experts, analysts, software project manager, designers/developers/implementers, integrators, testers, and quality assurance stakeholders).

4. Differences Between Patterns and Reference Architectures

Patterns can be related to other patterns in different ways [Buschmann et. al 2007]. As a reference architecture is a broader solution, in some cases, it is more fair to compare such architecture to a set of patterns instead to a single one. A Pattern Collection can be used to organize and classify patterns related to a common domain. A Pattern Compound, also known as Compound Pattern, can document a recurrent combination of patterns, which can be considered itself a pattern. Moreover, a pattern language, which focus on how patterns relate in a given domain, is a strongest relation that patterns can have, documenting a broader knowledge in that context. Despite the term "pattern language" is commonly used by the pattern community, there is no formal boundary that divides pattern collections from pattern languages. In this scenario, this paper refers to pattern language as a set of related patterns..

An important difference between a pattern and a reference architecture is that a pattern must be a recurrent solution that was already used on existing implementations. On the other hand, a reference architecture can propose new, innovative solutions in its structure, whose usage was not yet proven in existing softwares. As a consequence, a reference architecture is more suitable to propose a structure for new domains, which are not still well established. Moreover, for a domain that already has many applications implemented, it is interesting to establish such architecture, but it is necessary to identify the recurrent solutions used by these applications.

Patterns and reference architectures document solutions in different levels of granularity. While a pattern focus on a single problem and in a single recurrent solution, a reference architecture usually considers the target domain as a whole. Hence, a pattern language or other kinds of pattern grouping are closer to the solution scope of a reference architecture. However, even a pattern language does not have explicit requirements to fulfill and is not worried about completeness, it documents recurrent solutions that are already well established for the domain it focuses.

The domain of a pattern is related to its recurrency. It can be more general or more specific to a given domain, depending on the problem in which it is related to and the scope where the pattern mining was performed. On the other hand, a reference architecture is commonly focused on a single domain, containing patterns that can be appropriate for other architectures, too.

A pattern is documented based on a template composed by sections, which are at least its context, its problem, and its solution. Other common sections present the forces, consequences, alternatives for implementations, and examples of use. In a pattern language, the patterns usually have a standard structure. A reference architecture has a more comprehensive documentation, usually with many pages, including models, tables, texts, and in some rare cases implementation.

When we consider evolution, a fast way to improve a pattern language is by adding new pattern to it. Being a recurrent solution, this new pattern can complement the existing knowledge in different ways, either by offering an alternative solution or by adding a solution to a different problem. However, this addition should be done carefully, avoiding redundant parts and confusions. When evolving a reference architecture, a new version is released. This version can present minor or major additions, but it should be released as a whole.

Table 1 summarizes the differences between both of them. The column related to patterns presents characteristics for a single pattern and a pattern language, when applicable.

Characteristic	Single Pattern / Pattern Language	Reference Architecture
Information originality	Recurrent solution	Innovative solution or recurrent solution
Granularity	Single problem / Single problem for each pattern in a shared domain	Multiple related problems

Relation to application domain	Independent or dependent of the application domain	Dependent of the application domain in the most of the cases
Documentation	Pattern description, usually using a standard pattern template / A diagram relating the patterns	Many pages, including models, tables, texts, and even implementation
Update	New implementation strategies and variations / Additions of new patterns	Minor or major additions considering the whole document

Table 1: Summary of Differences between Patterns and Reference Architectures

5. From a Pattern Language to a Reference Architecture

The goal of this section is to go towards guidelines to create a reference architecture based on a pattern language. It is worth highlighting that when reference architectures are created, several information sources must be considered, including reference models, domain models, conceptual models, domain ontologies, and so on. In particular, our proposal explicitly explore the use of pattern languages and patterns for that. Reference architectures can be also built out from analysis/design patterns; moreover, other patterns, such as for security, safety, or reliability, can be combined into such architectures.

Our guidelines can also be followed to apply other types of pattern groupings such as patterns collections to contribute to the reference architecture building. It is important to highlight that a reference architecture does not necessarily need to be based on patterns, and even when it is, it can also use some ad-hoc solutions to problems in which the solution is not still consolidated in existing implementations.

As stated before, a pattern language is a set of related patterns that capture recurrent solutions for common problems in a given domain. Despite patterns can document relations and dependencies with other patterns in the language [Noble 1998], a pattern language does not present a single structure. This fact is important to create a reference architecture based on pattern language, presenting how the patterns participants can fit together in the same structure.

A pattern usually present components (often described in a "Participants" section), which interact to solve a target problem. In a pattern language, patterns handle problems in the same domain. Hence, it is common to have components that play the same role in different patterns of the language. Based on that, the structure can be unified by using the common components as points to bind the pattern solutions together.

When there is some patterns or groups of patterns that ends up without a connection, there can be some missing pattern on the pattern language. That can be a problem, because the participants need to be bound to a single structure in the reference architecture. One option is to perform some pattern mining on the existing systems to specifically search for the solution that binds the structure. If the solution used by the systems is not uniform, in other words, there is not a consolidated pattern, a new or a particular solution can be proposed and used in the reference architecture. Pattern solutions can also be complemented by new proposals or solutions inspired in more general patterns.

When a pattern language presents alternative patterns for the same problem, one of them should be chosen to be used in the reference architecture. Despite having a discussion on how to add this kind of variability on reference architectures [Nakagawa 2012B], it is not common to find alternative solution in existing architectures. An alternative to add both solutions is to define variabilities on the reference architecture, representing different implementations. However, supportting both solutions in the same structure might not represent faithfully the domain's needs, since only one of them should be used.

It is important to highlight that these ways to handle missing patterns or alternative solutions to define reference architectures from a pattern language need deeper investigations and validation. They are both a first initiative on how they can be handled.

An example of this initiative is the reference architecture for metadata based frameworks [Guerra et. al 2013A]. This reference architecture was created based on a pattern language [Guerra et. al 2013B] that documents eight patterns focused on recurrent solutions for metadata processing and reading used in this kind of framework. In this case, the patterns in the pattern language were enough to bind all the components referenced on the patterns.

The same happened in the Adaptive Object Models (AOM) domain. Initially, some patterns documented practices on how to create adaptive objects using metadata [Yoder and Foote 1998]. After that, it evolved to what was called an architectural style [Yoder and Johnson 2002], which is in some extent closer to a reference architecture. Since it is easier to add extensions on a set of patterns, future works focused on other aspects of the AOM architecture, such as graphical interface rendering [Welicki et. al 2007] and the creation of objects [Welicki et. al 2009]. Despite a new reference architecture was not defined nor the architectural style was redefined, since these new patterns were meant to be used with the initial AOM patterns, they are perfectly compatible with the documented AOM architectural style. Other similar examples are the reference architecture for service-based systems [Striker et. al 2010] and a security reference architecture for cloud systems [Fernandez et. al 2015].

6. Future Perspectives

This section focus on some future research perspectives that involve patterns and reference architectures focusing on how one can contribute to the creation of the other one.

6.1. Using Reference Architectures for Pattern Mining

Pattern mining activity consists in searching recurrent solutions in a target domain. For architectural design and, more generally, software patterns, this search can be conducted by using code inspection. Software documentation, such as diagrams and descriptions, can also be used in this activity.

This search can be focused on the problem, looking on software how they solve a particular issue. The pattern mining can be also based on the structure of several software related to the same domain, searching for common practices. In both activities, the reference architecture can be used as a source for the pattern mining. It can be compared to actual implementations to identify the its recurrency. A reference architecture that was already used on several projects can also be a source of information.

6.2. Pattern-based Method for Reference Architecture Creation

Aiming at exploring the benefits of using recurrent, well-tested solution during the construction of reference architectures, patterns become quite interesting to be included in ProSA-RA. In particular, Step RA-3 can be benefited by these patterns, what includes mainly design patterns and architectural patterns. However, depending on the reference architecture domain, other kinds of patterns can also be considered, such as security patterns, safety patterns, and testing patterns.

During the conduction of this step, there are several sources of patterns that can be used, such as patterns repositories, pattern collections or pattern languages related to the reference architecture domain, as well as patterns related to a problem domain that is being faced on the reference architecture (security, reliability, distribution, etc...). A particular source can be also specialists in patterns or domain specialists. The patterns can be used in the following activities:

- Identification of a set of candidate patterns. For this, requirements of the reference architecture identified in the last step must be considered;
- Selection of patterns that fit the needs of the reference architecture. Benefits and drawbacks of adopting each pattern must be investigated separately;
- Joint analysis of all patterns that are intended to be adopted;
- Decision of the Points of Application (PA), i.e., part/component of the reference architecture where each pattern will be applied and also when two or more patterns are concurrent in the same PA. This last issue characterizes the variation points of the reference architecture regarding pattern application; and

- Design of the reference architecture by creating architectural views and models that include the selected patterns.

Besides using patterns for these architectures, the use of pattern languages are also aligned with goals of such architectures.

When the search does not find a pattern for a given problem related to a reference architecture, an alternative is to conduct a pattern mining activity to find a pattern on existing systems. This pattern can be documented and stand on its own, and can be further be incorporated in the reference architecture.

7. Conclusions

Patterns, pattern languages, and reference architectures must be investigated in a complementary way to provide a more complete solution to the development of software systems. Initiatives in that direction can be already found and there are still good open research perspectives.

The use of patterns and pattern languages to create reference architecture could bring important contributions, such as reduction in time and efforts to design reference architectures and improvement in the reference architecture quality, as parts of such architecture adopt well-tested solutions.

It is important to highlight that both areas (patterns and reference architecture) have already achieved a consolidated status, with recognized advantages and solutions; therefore, it seems natural to investigate them together, contributing to the advance of the state-of-the-art of these areas.

Acknowledgments

This study is supported by São Paulo Research Foundation (FAPESP, Grant: 2014/02244-7 and 2015/16487-1), CNPq (Grant: 445562/2014-5) and Brazilian funding agency Capes (Grant: 2012/034). We would also like to thank Prof. Dr. Eduardo Fernandez from Florida Atlantic University (FAU) who served as a shepherd, providing us valuable comments that led us to a considerable improvement of the paper.

References

[Angelov, 2012] S. Angelov, P. Grefen, and D. Greefhorst, “A Framework for Analysis and Design of Software Reference Architectures,” Information and Software Technology, vol. 54, no. 4, pp. 417–431, 2012.

[Avgeriou, 2013] P. Avgeriou, “Describing, Instantiating and Evaluating a Reference Architecture: A Case Study,” Enterprise Architect Journal, Fawcette Technical Publications, Jun. 2003.

[Arsanjani, 2007] A. Arsanjani,L.J. Zhang, M. Ellis, A. Allam, K. Channabasavaiah, “S3: A Service-Oriented Reference Architecture”, IT Professional, vol. 9, no. 3, p. 10-17, IEEE Computer Society, 2007.

[Autosar, 2015] AUTOSAR, “AUTOSAR (AUTomotive Open System ARchitecture)”, [On-line], World Wide Web, 2015, Available in: http://www.autosar.org/ (Access in 07/30/2015).

[Bayer, 2004] J. Bayer, T. Forster, D. Ganesan, J. F. Girard, I. John, J. Knodel, R. Kolb, and D. Muthig, “Definition of Reference Architectures Based on Existing Systems,” Fraunhofer IESE, Tech. Rep. 034.04/E, 2004.

[Buschmann et. al, 2007] F. Buschmann, K. Henney, D. C. Schmidt. "Pattern-Oriented Software Architecture", vol. 5, On Patterns and Pattern Languages, Wiley, 2007.

[Cloutier, 2010] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, and M. Bone, “The Concept of Reference Architectures,” Systems Engineering, vol. 13, no. 1, pp. 14–27, 2010.

[Continua, 2015] CONTINUA HEALTH ALLIANCE, “Continua Health Alliance”, [On-line], World Wide Web, 2013, Available in: http://www.continuaalliance.org/ (Access in 07/30/2015).

[Dobrica, 2008] L. Dobrica and E. Niemela, “An Approach to Reference Architecture Design for Different Domains of Embedded Systems,” in SERP ’08, Las Vegas, USA, Jul. 2008, pp. 287–293.

[Fernandez et. al, 2015] E.B.Fernandez, R. Monge, and K. Hashizume, “Building a Security Reference Architecture for Cloud Systems”, Requirements Engineering, Jan. 2015, pp. 1-25

[Galster, 2011] M. Galster, P. Avgeriou, D. Weyns, T. Männistö, “Variability in Software Architecture: Current Practice and Challenges”, SIGSOFT Software Engineering Notes, vol. 36, no. 5, 2011, p. 30-32.

[Guerra et. al, 2013A] E. Guerra, F. Alves, U. Kulesza, C. Fernandes, "A Reference Architecture for Organizing the Internal Structure of Metadata-based Frameworks," Journal of Systems and Software, vol. 86, no. 5, May 2013, p. 1239-1256

[Guerra et. al, 2013B] E. Guerra, J. Souza, C. Fernandes. "Pattern Language for the Internal Structure of Metadata-Based Frameworks". In Transactions on Pattern Languages of Programming III, Lecture Notes in Computer Science Volume 7840, 2013, p. 55-110.

[Muller, 2008] MULLER G., “A Reference Architecture Primer”, [On-line], World Wide Web, 2008, Available in http://www.gaudisite.nl/ (Access in 07/30/2015).

[Nakagawa, 2012A] E. Y. Nakagawa, F. Oquendo, and M. Becker, “RAModel: A Reference Model of Reference Architectures,” in WICSA/ECSA ’12, Helsinki, Finland, 2012, pp. 297–301.

[Nakagawa 2012B] E.Y. Nakagawa. "Reference Architectures and Variability: Current Status and Future Perspectives," WICSA/ECSA '12, Helsinki, Finland, 2012, p. 159-162.

[Nakagawa, 2015] E.Y. Nakagawa, F. Oquendo, and J.C. Maldonado, "Reference Architectures", In: Oussalah, M.; Software Architecture: Principles, Techniques, and Tools, Ed. John Wiley & Sons, p. 101-122.

[Noble, 1998] J. Noble. "Classifying Relationships between Object-Oriented, Design Patterns," ASWEC 98, Adelaide, Australia, 1998, P. 98-107

[Papazoglou, 2008] M. P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, “Service-Oriented Computing: A Research Roadmap" International Journal of Cooperative Information Systems, v. 17, n. 2, p. 223–255, 2008.

[Oasis, 2008] OASIS, "Reference Architecture for Service Oriented Architecture Version 1.0," Report, OASIS Standard, April 2008.

[OSGI, 2015] OSGI ALLIANCE, “OSGi Alliance Specifications”, [On-line], World Wide Web, 2013, Available in http://www.osgi.org/Specifications/ (Access in 07/30/2015).

[Striker et. al, 2010] V. Stricker, K. Lauenroth, P. Corte, F. Gittler, S. De Panfilis, and K. Pohl, “Creating a Reference Architecture for Service-Based Systems – A Pattern-Based Approach,” in Towards the Future Internet - Emerging Trends from European Research, G. Tselentis, A. Galis, A. Gavras, S. Krco, V. Lotz, E. Simperl, B. Stiller, and T. Zahariadis, Eds. IOS Press, 2010.

[Universaal, 2015] UNIVERSAAL PROJECT, “The UniversAAL Reference Architecture," [Online], World Wide Web, 2013, Available in http://www.universaal.org/ (Access in 07/30/2015).

[Yoder and Foote, 1998] J. W. Yoder and B. Foote, "Metadata and Active Object Models,"PLoP ’98, Monticello, USA, 1998, p. 022-A22.

[Yoder and Johnson, 2002] J. W. Yoder and R. Johnson, "The Adaptive Object Model Architectural Style," in WICSA ’02, Montreal, Canada, 2002, p. 3-27.

[Welicki et. al, 2007] L. Welicki, R. Wirfs-Brock, and J. W. Yoder, "Rendering Patterns for Adaptive Object-Models,"PLoP ’07, Monticello, USA, 2007, p. 12:1-12:12.

[Welicki et. al, 2009] L. Welicki, R. Wirfs-Brock, and J. W. Yoder, "Adaptive Object Model Builder," in PLoP ’09, Chicago, USA, 2009, p. 4:1-4:8.