Regular Paper

The Concept of Reference Architectures

Robert Cloutier,[1,*] Gerrit Muller,[2,†] Dinesh Verma,[1] Roshanak Nilchiani,[1] Eirik Hole,[1] and Mary Bone[1]

[1]*School of Systems and Enterprises, Stevens Institute of Technology, Hoboken, NJ 07030*
[2]*Embedded Systems Institute, Eindhoven, Netherlands*

Received 4 March 2008; Revised 12 August 2008; Accepted 23 October 2008, after one or more revisions
Published online 22 January 2009 in Wiley InterScience (www.interscience.wiley.com)
DOI 10.1002/sys.20129

ABSTRACT

The concept of Reference Architectures is novel in the business world. However, many architects active in the creation of complex systems frequently use the term *Reference Architecture*. Yet, these experienced architects do not collectively have a consistent notion of what constitutes a Reference Architecture, what is the value of maintaining the Reference Architecture, what is the best approach to visualizing a Reference Architecture, what is the most appropriate level of abstraction, and how should an architect make use of the Reference Architecture in their work? This paper examines current Reference Architectures and the driving forces behind development of them to come to a collective conclusion on what a Reference Architecture should truly be. It will be shown that a Reference Architecture captures the accumulated architectural knowledge of thousands man-years of work. This knowledge ranges from why (market segmentation, value chain, customer key drivers, application), what (systems, key performance parameters, system interfaces, functionality, variability), to how (design views and diagrams, essential design patterns, main concepts). The purpose of the Reference Architecture is to provide guidance for future developments. The Reference Architecture incorporates the vision and strategy for the future. The Reference Architecture is a reference for the hundreds of teams related to ongoing developments. By providing this reference all these teams have a shared baseline of why, what and how. It is the authors' goal that this paper will facilitate further research in the concepts and ideas presented herein. © 2009 Wiley Periodicals, Inc. Syst Eng 13: 14–27, 2010

Key words: reference architecture; pattern; system architecture; architecture framework; interoperability

1. INTRODUCTION

There is a fair amount of literature on Reference Architectures, though much of it is for the software engineering audience, inconsistent, and not systems engineering oriented. Although the idea of a Reference Architecture has become a known concept, the form that Reference Architecture will take is still not solidified. The lack of maturity of the term "Reference Architecture" can be seen by the small number of articles and books that address this specific topic from a systems engineering perspective.

This paper will help to solidify the concept of a Reference Architecture from a systems engineering point of view. The authors will also attempt to lay down a foundation for how a Reference Architecture can be identified. While discussing how to identify a Reference Architecture, a brief discussion on how to recognize patterns will be given. The idea of patterns plays a large role in Reference Architecture. One definition states a Reference Architecture is [Reed, 2002]:

> According to RUP (Rational Unified Process), a Reference Architecture: …is, in essence, a predefined architectural pattern, or set of patterns, possible partially or completely instan-

*Author to whom all correspondence should be addressed (e-mail: robert.cloutier@stevens.edu).
† Present address: Buskerud University College, Kongsberg, Norway

tiated, designed, and proven for use in particular business and technical contexts, together with supporting artifacts to enable their use. Often, these artifacts are harvested from previous projects.

As mentioned earlier, this is a Reference Architecture definition from a software perspective and does not represent the "whole of the system." However, it does bring together the notion of Reference Architectures, patterns and reuse.

This paper was inspired by the System Architecture Forum,[1] co-hosted by Stevens Institute and the Embedded Systems Institute twice a year. Each forum has a theme to be discussed by the attending system architects. Much of this paper documents the discussion and findings from one such meeting, where the theme was Reference Architectures: What are they, and what is the value of a Reference Architecture? This meeting was attended by several architects from the defense domain and the commercial equipment domain who offered presentations based on their experiences using and/or creating Reference Architectures.

This paper begins by exploring current Reference Architectures and definitions. Then, it moves on to explore what the driving factors are for developing these Reference Architectures. The paper will next go into describing how the idea of a Reference Architecture can be solidified. During this discussion the concepts of patterns are explored in detail and the concept of abstraction is described. The mining and discovery of patterns, and the implicit and explicit knowledge in an architecture are discussed as essential topics in Reference Architecture. The importance of transforming implicit knowledge to explicit knowledge in system architecture is discussed. Patterns are presented as a powerful tool to capture the implicit knowledge for Reference Architectures. Finally, the paper will end with a summary and recommendations for continued study of this topic.

2. DEFINITIONS

A number of terms related to system architecture are in use today. It often becomes difficult to communicate using terms that are used imprecisely and with varying frames of reference. One such word is "architecture." The online Merriam-Webster dictionary states that it is[2]:

> 1: the art or science of building; specifically: the art or practice of designing and building structures and especially habitable ones
>
> 2 a: formation or construction resulting from or as if from a conscious act <the architecture of the garden>
>
> b: a unifying or coherent form or structure <the novel lacks architecture>

Today, even political speeches and political comebacks are said to have an architect and an architecture. However, in engineering context, if one were to apply the definiton cited above, the art or science of building, then one can easily extend architecture to the designing and building of any complex system. Where possible, this paper will use terminology that has been broadly vetted and adopted by professional organizations or authored works and, therefore, stands the test of time to provide the basis for clarity for the remainder of this paper. The defintions chosen here are not a complete list of terms that will be used in this paper but a selection of terms that the authors felt needed to be explicity stated so that no confusion would arise later as to their usage in this paper:

Architect: The person, team, or organization responsible for systems architecture [IEEE 1471, 2000].

Architecture: The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution [IEEE 1471, 2000].

Architecture Framework: An *architecture framework* provides guidance and rules for structuring, classifying, and organizing architectures [DoDAF, 2007].

Design Pattern: A design pattern systematically names, motivates, and explains a general design that addresses a recurring design problem in a system. It describes the problem, the solution, when to apply the solution, and its consequences. It also gives implementation hints and examples. The solution is customized and implemented to solve the problem in a particular context [Gamma et al., 1995].

Explicit Knowledge: information that has been shared and even possibly documented, for the use of others.

Implicit Knowledge: information known by an individual, hidden in the realizations, or spread over a group of individuals, but not shared or documented for the use of others.

Perspective: the posture from which an observer may be analyzing a problem. For instance, a system may be analyzed from a developer's perspective, a systems engineer's perspective, or even a lifecycle perspective.

System: A system is a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents, that is, all things required to produce systems-level results. The results include system level qualities, properties, characteristics, functions, behavior and performance. The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts; that is, how they are interconnected [Maier and Rechtin, 2000].

[1]System Architecture Forum is a loose, multinational collection of architects from various business enterprises—large and small. Representatives come from defense industries, medical system designers, communications, and research institutes. The goal of the System Architecture forum is to share architecture best practices and emerging research among the participants. http://www.architectingforum.org/

[2]http://www.merriam-webster.com/dictionary/architecture

Systems Engineer: An engineer that practices an interdisciplinary approach and means to enable the realization of successful systems [INCOSE, 2000].

View: A representation of a whole system from the perspective of a related set of concerns [IEEE 1471, 2000].

Viewpoint: A specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis [IEEE 1471, 2000].

3. THE CURRENT VIEW OF A REFERENCE ARCHITECTURE

Although there are many instances in technical literature and documents where the term "Reference Architecture" is used, it is done so without a solid definition. For this reason the term "Reference Architecture" has became a term to mean many things to different people. For instance, Sun Computer[3] states that:

> Sun's Reference Architectures have been designed, tested, tuned and documented, so you can reduce the complexity, costs, and risks of deploying new technology in your enterprise. Before choosing to implement a Reference Architecture, you can test a proof-of-concept system at a Sun Solution Center. Sun's Reference Architectures combine:
>
> - A documented multi-tiered architecture
> - Recommended technology products from Sun and other vendors
> - Architecture, sizing, and implementation guides

Another company, Burton Group Research and Advisory,[4] has a product they have named "Reference Architecture," which is "a proprietary knowledgebase built on Burton Group's research and real-world experience from hundreds of global enterprise consulting engagements." This tool claims it incorporates the following principles: high-level statements about the IT infrastructure that tie back to business goals, organizational, environmental, and political context, and drives Technical Positions and Templates.

BEA Systems provides a document that promises "a worked design of an enterprise-wide service oriented architecture (SOA) implementation, with detailed architecture diagrams, component descriptions, detailed requirements, design patterns, opinions about standards, patterns on regulation compliance, standards templates, and potential code assets from members".[5]

From these descriptions, one can see how varied the definition of "Reference Architecture" is used. While in the Sun's Reference Architecture their focus is on technology and implementation, the Burton Group Research and Advisory Reference Architecture is more focused on business goals. Yet, in the third, BEA Systems, Reference Architecture focuses on "detailed" requirements and "detailed" architecture. These examples could go on and on with how the term "Reference Architecture" is used differently by people, either within the same industry or not.

These split views of a Reference Architecture can be further seen in today's practice, where the term Reference Architectures is used for very different entities. The IT world uses the term Reference Architecture often for the infrastructural concepts, for example, for layering, communication and persistency. See, for example, Simmons [2005] and Reed [2002]. The definition[6] provided by Wikipedia starts with general terminology, and then narrows to IT.[7]

At Philips Medical System the term Reference Architecture has been used for a much higher level domain understanding. In Muller [2004] a Reference Architecture is discussed that provides guidelines for function allocation over products based on technology, life-cycle, safety and information characteristics. As shown here, a Reference Architecture is created with a certain scope in mind, e.g., a domain of a set of applications. In this scope the Reference Architecture links to relevant standards, legislation, domain constraints, and mandatory frameworks.

To further explore how Reference Architecture is currently viewed, this paper will discuss a couple of existing Reference Architectures in various domains and highlight some key characteristics of each Reference Architecture. No judgment is made in this section on whether these examples are good examples of a Reference Architecture. The examples are provided purely as a baseline for how different industries currently use and define a Reference Architecture. Although in Section 5 the authors will address the concept of creating a well-defined and valued Reference Architecture.

3.1. Reference Architecture for Space Data Systems

This is a draft recommended practice prepared by the Consultative Committee for Space Data Systems, dated January 2007 [CCSDS, 2007]. It is being prepared for NASA. Some of the member agencies participating include Agenzia Spaziale Italiana (ASI)/Italy, Canadian Space Agency (CSA)/Canada, European Space Agency (ESA)/Europe, and the Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil. The document is 107 pages in length and opens with a scope statement:

> This document presents a Reference Architecture for Space Data Systems (RASDS). The RASDS is intended to provide a standardized approach for description of data system architectures and high-level designs, which individual Consultative Committee for Space Data Systems (CCSDS) working groups may use within CCSDS. This approach is aligned with best current practices in the fields of system and software architecture and is specifically adapted for the space domain.

[3] http://www.sun.com/service/refarch/

[4] http://www.burtongroup.com/research_consulting/ref_architecture.asp

[5] http://dev2dev.bea.com/2006/09/SOAPGPart2.pdf

[6] A Reference Architecture provides a proven template solution for an architecture for a particular domain. It also provides a common vocabulary with which to discuss implementations, often with the aim to stress commonality.

[7] A Reference Architecture often consists of a list of functions and some indication of their interfaces (or APIs) and interactions with each other and with functions located outside of the scope of the Reference Architecture.

> While it is intended for use within CCSDS it is also suitable for use by mission and project design teams, to describe system architectures and designs within the space domain.

More importantly, under the rationale section of the Space Systems Reference Architecture, the Reference Architecture cites numerous international standards, to include the ISO Reference Model for Open Distributed Processing [ISO/IEC 10746-1, 2004], the Recommended Practice for Architectural Descriptions of Software-Intensive Systems [IEEE 1471, 2000] and Standard for Application and Management of the System Engineering Process [IEEE 1220, 2005]. Then, the Reference Architecture goes on to state that those standards all assume the elements of systems are fixed in place and are in continuous communication over what are nominally error free communications channels that suffer only occasional disruptions and that assumption is not valid for space data systems.

3.2. An Information Technology Security Architecture for the State of Arizona

This Security Architecture is part of a broader effort to provide a statewide roadmap of technology to meet the State of Arizona's mission. In the executive summary (see State of Arizona), the Security Architecture cites the higher-level guidance for Reference Architectures provided by the GITA (a statewide IT agency). These guidelines may provide some good insight into the purpose of any Reference Architecture under development:

The following ten general principles provide the framework for defining the baseline and target architectures within the technical domains:

1. Architectures must be appropriately scoped, planned, and defined based on the intended use of the architecture.
2. Architectures must be compliant with the law as expressed by legislative mandates, executive orders, State and Federal regulations.
3. Architectures facilitate change.
4. Enterprise architectures must reflect the Governor's Strategic Plan, The Statewide IT Strategic Plan and the Agency's Three-year IT Plan.
5. Architectures continuously change and require transition.
6. Target architectures should project no more than 3 to 5 years in the future.
7. Architectures provide for a standardization of business processes and common operating IT environments.
8. Architecture products are only as good as the data collected from subject matter experts and domain owners.
9. Architectures minimize the burden of data collection, streamline data storage, and enhance data access.
10. Target architectures are used to control the growth of technical diversity.

4. DRIVING FORCES FOR A REFERENCE ARCHITECTURE

From the previous sections, it is clear that companies and industries are using what they call "Reference Architectures," although the examples given above show that currently Reference Architecture can have varying meaning. While reviewing these examples of Reference Architectures, one can construct a full mental picture of what constitutes a Reference Architecture. Based on that mental picture, the following working definition, as put forth by the authors, for a Reference Architecture might stand scrutiny:

> Reference Architectures capture the essence of existing architectures, and the vision of future needs and evolution to provide guidance to assist in developing new system architectures.

This definition attempts to take into account all the driving forces for identifying a Reference Architecture which were used to develop the examples in this paper and many others that were reviewed while studying the topic of Reference Architecture. This section will identify those driving forces behind the development of current Reference Architectures to give more solidity to the definition just put forth, and the driving forces that the authors believe are lacking from current Reference Architectures but are imperative for the creation of a good Reference Architecture will be discussed and justified.

Many of these driving forces were identified through a chaotic discussion at the System Architectures Forum and were captured in a structured graph detailing the main objectives of a Reference Architecture (see Fig. 1). While many of these objectives are architecture objectives, they are even more relevant to reference architectures due to the increased scope of reference architectures. Figure 1 captures the concurrent trends of increased complexity and increased integration dynamics, and the objectives of Reference Architectures as reaction to these trends. These main objectives are achieved by more detailed objectives of Reference Architectures, shown at the right hand side of the graph of objectives.

4.1. The "Multi" Effect

During the aforementioned System Architecture Forum, it became clear that in all domains there were two simultaneous emerging trends that drive the development of a Reference Architecture:

- Increasing complexity, scope, and size of the system of interest, its context, and the organizations creating the system
- Increasing dynamics and integration: shorter time to market, more interoperability, rapid changes, and adaptations in the field.

These trends cause a transition from "simple" closed system creation to distributed open system creation and evolution. In the simple and closed situation, a system could be created at one location, by one vendor, in one organizational entity. Many of today's systems are developed as distributed

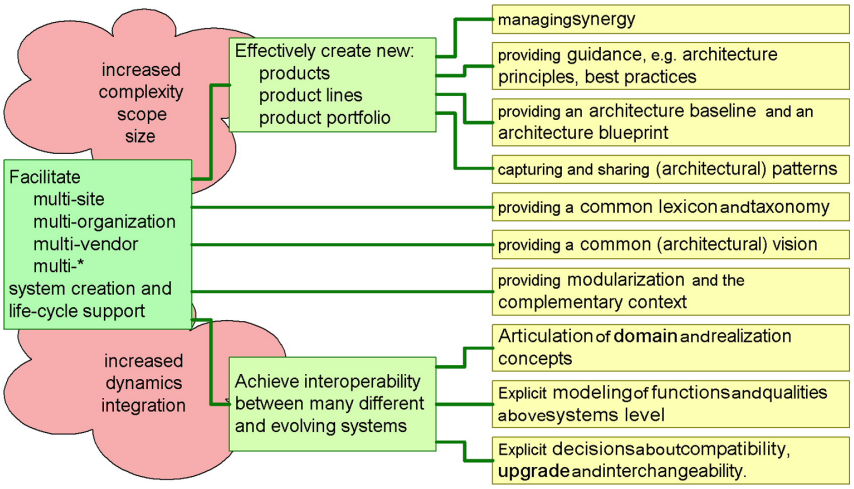


Figure 1. Graph of objectives of Reference Architectures. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

open development at multiple locations ("multisite"), by multiple vendors, across multiple organizations. The aggregate size of the developing organizations typically is in the order of hundreds or thousands of people working in teams of 100 or more team members that may be divided over 10 or more locations. These numbers are representative of many defense, telecommunications, health care, and semiconductor teams.

Refer back to Figure 1 and note the use of the word "multi." This was due to the multiplicity not being limited to organizations, vendors, and locations. Systems also become more multi-domain (e.g., security has military as well as civil applications), multiapplication (e.g., electron microscopes are used for metrology in high volume applications and for material analysis in low volume applications), multicultural (global application, but customized for local cultural aspects) development and manufacturing is based more often on multisourcing, and so on.

Reference Architectures start to appear in organizations where the multiplicity reaches a critical mass triggering a need to facilitate product creation and life-cycle support in this distributed open world. The Reference Architecture provides:

- a common lexicon and taxonomy
- a common (architectural) vision
- modularization and the complementary context.

The common lexicon and taxonomy facilitates communication across the multiple dimensions. The common (architectural) vision focuses and aligns efforts of multiple peoples and teams. Modularization helps to divide the effort, where the context information ensures later integration.

4.2. Effective Creation of Products, Products Lines, and Product Portfolios

Another driving factor for the creation of a Reference Architecture is to improve the effectiveness of creating products, product lines and product portfolios by:

- managing synergy
- providing guidance, e.g., architecture principles, best practices
- providing an architecture baseline and an architecture blueprint
- capturing and sharing (architectural) patterns.

Managing synergy is often the main goal of Reference Architectures from managerial perspective. It should be noted that maximization of synergy is *not* the goal of Reference Architectures. However, a good Reference Architecture helps in understanding where synergy can be harvested effectively and where harvesting of synergy might backfire. The insight that harvesting synergy is not always trivial has been formulated by Doug McIlroy [1968] at the 1968 NATO Conference about Software Engineering.

Reflection of experiences can be captured in architecture principles and best practices. This condensed, somewhat abstract, know-how provides guidance to later developments, hopefully preventing the reoccurrence of bad experiences over and over again. More concrete know-how can be mined by looking for architectural patterns. A pattern is a well working solution for a common problem that is described in terms of what circumstances and context this solution is appropriate.

The effectiveness is also improved by providing an architecture baseline, a shared starting point to discuss future changes and extensions. The Reference Architecture serves as an architecture pattern for future architectures. This helps prevent the reinvention and revalidation of solutions for already solved problems.

4.3. Achieving Interoperability between Many Different and Evolving Systems

In this "multi" world (e.g., multiple vendors, multiple contractors) *interoperability* determines the usability, perform-

ance, and dependability of user level applications. Reference Architectures are used to improve interoperability by:

- Articulation of domain and realization concepts
- Explicit modeling of functions and qualities above systems level
- Explicit decisions about compatibility, upgrade, and interchangeability.

Decreased integration cost and time might also be an objective of Reference Architectures. Note that all interoperability considerations are also applicable to reduction of integration cost and time. Note also that for reuse to be effective, it is required that integration effort must be small.

4.4. Mission, Vision, Strategy

A Reference Architecture is strongly linked to company (or consortium, e.g., Mobile Industry Processor Interface) mission, vision, and strategy. The strategy determines what multidimensions have to be addressed, what the scope of the Reference Architecture is, what means, such as synergy, are available to realize mission and vision. In fact, a Reference Architecture is an elaboration of mission, vision and strategy, as shown in Figure 2.

> ***Reference Architecture Principle 1:*** *A Reference Architecture is an elaboration of company (or consortium) mission, vision, and strategy. Such Reference Architecture facilitates a shared understanding across multiple products, organizations, and disciplines about the current architecture and the vision on the future direction.*

4.5. Linking Past and Future

A Reference Architecture can also be developed to facilitate a shared understanding across multiple products, organizations, and disciplines about current architecture(s) and future directions. Proven architectures of past and existing products are transformed in a Reference Architecture. However, the purpose of the Reference Architecture is future oriented—that is, to provide guidance for future implementations. This forward looking application is performed by initially using the Reference Architecture as the basis for future products, and then modifying the actual new product implementation by applying the new product mission, vision, and strategy to the wisdom of the past.

5. REFERENCE ARCHITECTURE UNFOLDED

Many architecture descriptions that are labeled as a Reference Architecture only describe the technical architecture. The participants of the System Architecture Forum meeting mentioned earlier in this paper concluded that a Reference Architecture should address:

- Technical architecture
- Business architecture
- Customer context.

In practice, business architecture and customer context are often missing (see Rosen [2007]). As a consequence, these technical Reference Architectures represent solutions for unspecified problems in unspecified contexts. Figure 3 shows the *business architecture*, the *technical architecture,* and the *customer context* as partially overlapping. The common denominator is the requirement or black box specification level, where the features and functions are modeled in a product independent way. In practice, business architecture and customer context are often missing (see Rosen [2007]). As a consequence these technical Reference Architectures represent solutions for unspecified problems in unspecified contexts. Figure 3 shows the *business architecture*, the *technical architecture,* and the *customer context* as partially overlapping. The only common denominator is the requirements or black box specification level, where the features and functions are modeled in a product independent way. The technical architecture provides solutions in technology, captured as design patterns. The business models and life cycle considerations in the business architecture guide decisions in the technical domain. The same holds for the customer context, where processes in the customer enterprise and user considerations will provide this guidance. Guidance from the Ref-

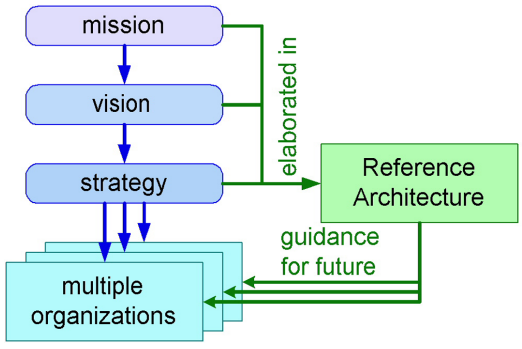


Figure 2. Business architecture, technical architecture and customer context together form the Reference Architecture. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

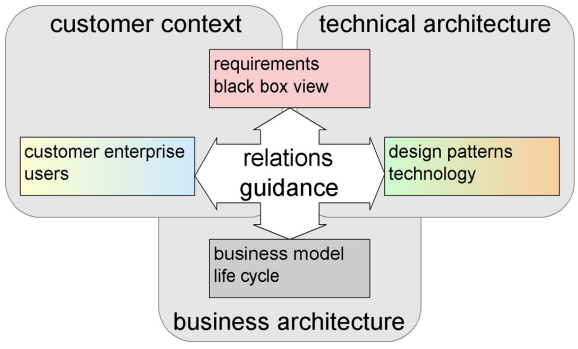


Figure 3. Business architecture, technical architecture and customer context together form the Reference Architecture. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

erence Architecture is largely based on the explicit understanding of the relations between the *business architecture*, the *technical architecture,* and the *customer context*.

The level of abstraction of a Reference Architecture makes it more difficult to understand their role. Figure 4 shows the instantiations that are needed to transform an abstract Reference Architecture into actual systems. The first step is to instantiate a system architecture based on the Reference Architecture. This system architecture is used to design and engineer the system, resulting in engineering documentation that describes how the system can actually be ordered, assembled and tested. This also demonstrated the difference between system architecture and Reference Architecture. A System Architecture is focused on a limited class of systems, both in product positioning as well as positioning in time. In comparison a Reference Architecture is much broader in scope, capturing the commonalities and essence. As a consequence, Reference Architecture may ignore many specific variations that serve well-defined market needs but that don't provide much reusable insights.

From Figure 4 note that the creation and evolution of Reference Architectures is strongly feedback based. Field feedback from actual systems results in updates of the engineering documentation. The design and engineering effort provides constraints on architectures, but also opens opportunities. Finally the Reference Architecture itself is largely a mining and extraction effort of existing architectures (whether the mining is done from implicit or explicit knowledge of relevant architectures).

The reuse or asset sharing dimension plays a role besides the instantiation dimension. If a product family is created, then we will instantiate a family architecture from the Reference Architecture. A family architecture describes the members of the product family and the mechanisms in the family to specialize members into the desired direction. The family architecture also describes the synergy within the product family and the associated rules for design, such as standardization. The shared assets often get a lot of focus, resulting in an architecture describing the shared assets (or platform).

In practice the term Reference Architecture is sometimes incorrectly used for family architecture and/or shared assets architecture. The role of a Reference Architecture is related to the instantiation flow as shown in Figure 4. Some people see the Reference Architecture as an *asset* capturing generalized knowledge of systems. This view of Reference Architectures positions the Reference Architecture in the area of knowledge management. This specification *asset* is seen as essential to effectively deploy a reuse strategy of actual design artifacts. As discussed earlier in the Reference Architecture contributes to communication effectiveness and provides guidance for future architecture instantiations.

5.1. Uncovering and Documenting Reference Architectures

There are common elements in each of the example Reference Architectures. Based on the small sampling, some of the common elements include: business purpose, standards, guidance for implementing, and roadmap. The Reference Architecture captures previous experience through mining and generalizing existing architectures; this will be discussed at more depth in the next section regarding Pattern Concepts. To be of value for future architectures, a Reference Architecture is based on *proven* concepts. The validation of concepts in Reference Architectures is often derived from preceding architectures. Especially in cases where disruptive technologies or innovative applications are introduced, it is challenging to have sufficient proof for a Reference Architecture. In these cases Reference Implementations and prototyping and an incremental approach might be an alternative for validation and proof. Note that flaws in Reference Architectures propagate to multiple architectures and actual systems and may damage or even destroy in that way entire enterprises.

Reference Architecture Principle 2*: A Reference Architecture is based on concepts proven in practice. Most often preceding architectures are mined for these proven concepts. For architecture renovation and innovation validation and*

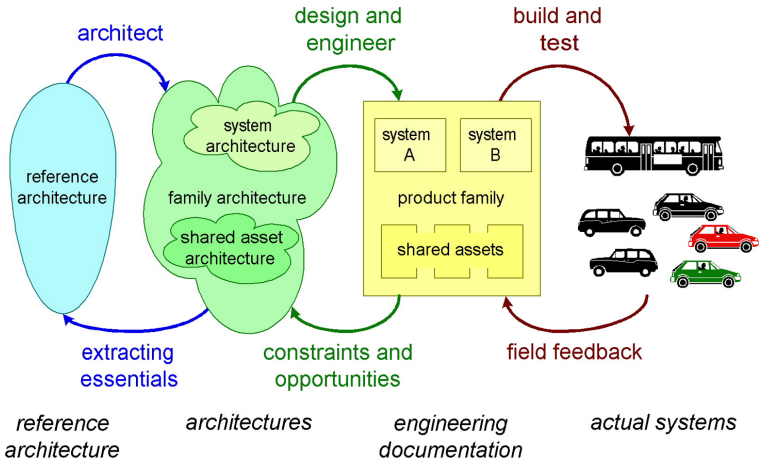


Figure 4. Reference Architectures are very abstract. In several instantiation steps a Reference Architecture is transformed into an actual system(s). [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

proof can be based on reference implementations and prototyping.

The future value of Reference Architecture depends on the vision going into it. This vision is based on (future) customer and business needs. These needs are explored and analyzed to be transformed into future requirements for the product portfolio. Figure 6 (subsection 5.3) shows this flow of proven concepts and known problems from existing architectures and vision derived from needs into the Reference Architecture. The Reference Architecture guides the evolution of existing architectures and influences the customers and business, which triggers new changes in their needs. Architectures, needs and Reference Architectures evolve continuously. Often, this is handled as a function of the architecture management process. Evolution of the Reference Architecture is managed through versioning of the published reference architectures.

5.1.1. Pattern Concepts

From Figure 5 one of the inputs into a Reference Architecture are architecture patterns. This discussion would therefore not be complete without discussing the concept of patterns.

While patterns are not a new concept and have been used by a number of engineering disciplines for many years, the concept was recently extended to systems engineering and complex systems. The system architecture pattern is defined as a high-level structure, appropriate to the design of the major components of a system, expressing the relation between the context, a problem, and a solution, documenting attributes, and usage guidance. They are time-proven in solving problems similar in nature to the problem under consideration. We will return to this definition later when we discuss Reference Architectures [Cloutier, 2006].

One of the key concepts for any successful pattern is that of separation. This concept allows the pattern documenter to separate the idea from the reality, capturing more generalized concepts when documenting the pattern. Another key concept is abstraction—the notion of removing detail from something complex to make it simpler to understand. Said differently, abstraction is the extraction of the "essence" to make something that is complex, more easily understood. While extracting the essence and removing the less relevant details, the art of abstraction is to keep the essential details to convey the true meaning. Many poor architects indeed simply leave out details and create "empty" abstractions.

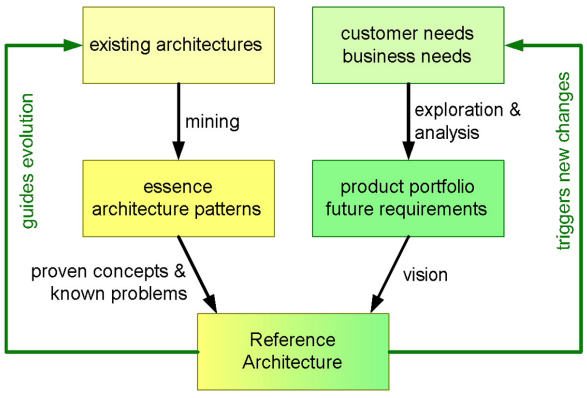


Figure 5. Inputs of a Reference Architecture. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

For instance, if one were to say they are talking on the phone, the concept of phone has been generalized. The listener does not know, nor care, whether they are talking on a portable phone, a cell phone, or a satellite phone. The details of what kind of phone have been abstracted out of the description of the phone call.

Abstraction is a very useful concept in describing complex constructs. However, in the practice of architecture, the challenge is finding the appropriate level of abstraction to describe the system being architected. Abstraction might be thought of as a continuum with multiple levels as the description is further refined. Therefore, the appropriate level of abstraction for a pattern is that level where the details of design can be implemented in a number of ways that still adheres to the intent of the pattern, and yet the pattern provides significant guidance on how that aspect should be architected.

Another valuable pattern concept within the software community is the rule of three, first identified by Ralph Johnson, which states that a pattern is not a pattern unless there are at least three independent, observable applications utilizing the proposed pattern as part of the solution. Brad Appleton, in his Web-based treatise "Patterns and Software: Essential Concepts and Terminology," states that there are some in the software patterns community that believe it is inappropriate to identify a solution as a pattern until that proposed pattern is vetted, or scrutinized, by others [Appleton, 2000]. It is interesting to note that some pattern practitioners actually feel it is inappropriate to decisively call something a pattern until it has undergone some degree of scrutiny or "peer review" by others. This scrutiny should give the user of the pattern some confidence in the completeness and appropriateness of the pattern, which may also reduce risk in a system employing the applied pattern. Once a pattern is identified and believed to be of use in the future, it should be documented.

If a pattern must be observed in three or more applications, then patterns cannot be invented. They "are not created from a blank page; they are mined" [Hanmer and Kocan, 2004]. When discussing design patterns, over time, similar designs are arrived at independently by different designers on various projects. As it becomes apparent that the same design elements exist in multiple designs, the design can be studied and documented to encourage reuse. This reuse "prevents the reinvention of the same concept, and provides a vocabulary for the design concepts that a project can share" [Sanz and Zalewski, 2003]. However, until patterns are mined and made explicit, they cannot be used widely and consciously. Note that many designers use a rich collection of patterns implicitly.

5.1.2. Implicit and Explicit Architectural Knowledge

Though one can attain advanced degrees in systems engineering, there are many systems engineers today that do not have a formal systems engineering degree. Instead, they have acquired experience and knowledge to be called a systems engineer by working on a number of development projects and by being able to think about how parts of the system

interact, both positively and negatively, with one another. Within industry and government, a growing number of practicing architects have acquired considerable architecture expertise via formal or informal mentoring and experience in the work environment. This corporate systems knowledge is captured in explicit ways through mediums such as handbooks, lessons learned repositories, templates and tools, methods and practices, and metrics and measures.

5.1.2.1. Implicit Knowledge. A significant component of this corporate knowledge, however, is implicit, undocumented, and largely represented through the technical leaders within an organization. This implicit knowledge is useful to others only if it is shared in a manner that allows its application. The holder of that implicit knowledge may become a bottleneck in applying systems experience on current or future projects [Hole, 2005]. The same holds true for knowledge that is reused from project to project in the form of undocumented or informal patterns.

The real power and value of a pattern is derived only if it can be packaged for use by others. If the pattern is not documented (written down), it may increase the possibility that the complete pattern will not be implemented the next time it is used—something may be forgotten or inadvertently left out. Formalizing patterns through documentation helps to ensure their usage by others, their management as assets, their application in an appropriate manner, and that they are improved over time.

5.1.2.2. Explicit Knowledge. Hahsler [2004] may be the only documented, quantitative source in documenting that the use of patterns improved communications between members of the architecture and design teams though patterns and heuristics are cited as serving to improve communications between teams [Rechtin, 1991; Maier and Rechtin, 2000]. Hahsler also found that patterns facilitated the application of sound architectural concepts and implementations. The sound architectural concepts identified by Hahsler were the software patterns documented by Gamma et al. [1995]. As the discipline of systems architecting assumes the challenge of developing increasingly complex systems, there is a need for a common lexicon between systems architects. Describing architectures in the context of known and understood patterns should foster better and more consistent understanding across the many stakeholder communities. Systems architecture patterns may also enable implementation of common design features across systems (reuse) leading to enhanced R&D efficiency, and lower ownership costs through reduced efforts with regard to system testing, integration, and maintenance.

5.1.3. Capturing Implicit Knowledge with Patterns

Patterns offer a formal method of documentation to capture aspects of such knowledge. If a pattern exists only in the form of implicit knowledge, it is not accessible by others and cannot be used by others without some form of repeated storytelling to convey the pattern to others by the holder of this knowledge [Cloutier and Verma, 2006].

While a pattern is reusable by the person who first recognizes it, the real power and value of a pattern is derived only if it can be packaged for use by others. Patterns can be transferred through examples, such as storytelling, or through informal or formal forms of documentation. Storytelling is one of the oldest and most proven methods of know-how transfer. The disadvantage of working by example is that concepts, and therefore the more fundamental know-how is not transferred. Contemporary examples of transferring by example are the so-called cookbooks; see, for example, Martelli and Ascher [2005]. This can be remedied by making the concepts more explicit. Informal documentation is appropriate for a broad set of stakeholders because the threshold to access the information is low.

More formal forms of documentation are appropriate in more narrow groups of stakeholders that share the same formalisms, because the information can be transferred more accurately and reliably. If the pattern is not formally documented, it may increase the possibility that the complete pattern will not be implemented the next time it is used—something may be forgotten or inadvertently left out. Formalizing patterns through documentation helps to ensure their usage by others, their management as assets their application in an appropriate manner, and their improvement over time.

5.1.4. From Patterns to Reference Architectures

One of the goals of Reference Architectures is to capture implicit knowledge to facilitate the development of news products and product families. Patterns are one of the means to document Reference Architectures, since patterns are well recognized for their capability to make implicit knowledge explicit. As stated in 5.1.2.1, patterns are only valuable if they can be used by others, while 5.1.2.2 emphasizes the communication value of patterns, where communication is also one of the main purposes of Reference Architectures. Also note that we use the concept of a pattern in Systems Engineering in a broader sense than the notion of pattern in Software Engineering. The common denominator is that patterns link common problems, to proven solutions providing a rationale related to the circumstances.

5.2. Aspects of the Reference Architecture

Another input into a Reference Architecture from Figure 5 is the essence of the architecture, the most significant and relevant aspects. A Reference Architecture can be supported by more detailed models, API's, standards, etc. The challenge is to create an easily readable, accessible Reference Architecture that at the same time is nonambiguous and effective for all stakeholders.

The size of Reference Architecture is domain and objective dependent. Attempts to find a metric for the size resulted in lots of discussions at the Systems Architecture Forum. Many existing Reference Architectures are very big (hundreds or even thousands of pages). The risk of big Reference Architectures is that the essence is hidden instead of highlighted. A countermeasure for big Reference Architectures is to decompose it into a core Reference Architecture and more detailed models, interfaces, standards, etc. Also, navigational support can improve accessibility.

Reference Architectures are often associated with big and inflexible frameworks; see, for example, the nice contribution in Martin [2003].

Content and structure of Reference Architectures is determined by domains and objectives. Typically a limited set

(about 6) of aspects or cross cutting concerns or attributes are captured in the Reference Architecture. For example, one of the presented Reference Architectures captures the following 6 aspects:

- Timeliness/responsiveness of system (real time vs. off line)
- Security considerations
- Degree of automation (machine vs. man)
- Legacy retention/interoperability and integration/test
- Life cycle concerns (Costs, "ilities")
- COTS content.

Domain-specific tensions between cross cutting concerns determine the specific selection of architecture patterns.

5.3. Applying and Adapting Reference Architectures

Reference Architectures must be actively managed by the organization to ensure that they are updated based on the evolution of the governing standards, emerging technology, changing stakeholder requirements and maturing business. Figure 6 attempts to capture the development cycle of a Reference Architecture.

Once developed, a Reference Architecture has a life-cycle and *maturity*. Reference Architectures should evolve continuously and must be actively maintained and re-factored. Finally, Architecture Management and Governance involves the total set of activities from existing architectures to Reference Architecture's patterns and to the creation of new architecture implementations.

From Figure 6 one can also see the relationship between the architecture content documented using a Reference Architecture and Architecture Framework, e.g., DoDAF, MoDAF, or Zachman [1987]. The Architecture Framework captures knowledge about the methodology to create and document architectures. The methodology knowledge ranges from views (purpose, stakeholders, concerns), representations (e.g., class diagrams, activity diagrams), tools, to methods (decomposition in tasks, ordering, and guidelines). These frameworks make a step in the meta-direction: "How to architect?" An architecture framework is comparable to a document template: It provides rules and guidelines for layout and style, and it might provide some decomposition and ordering of content. However, templates don't provide actual content. In contrast, Reference Architectures heavily focus on content. Reference architectures may use architecture frameworks, as seen in Figure 6, but it might well be that a Reference Architecture captures some highly domain specific diagrams not present in any of the more general purpose architecture frameworks.

5.4. Managing the Reference Architecture

A Reference Architecture must be accessible and understandable for multiple stakeholders from engineers to business managers and customers. Therefore, the Reference Architecture must be concrete and provide specific information. The challenge is to create a Reference Architecture that is generic for multiple architectures and that is concrete and contains specific information at the same time.

This may seem in conflict with the goal of a reusable Reference Architecture, but current examples that are similar to this can be found in the work-flow for brain examinations in MRI scanners which are very specific and concrete yet they are reused over many generations of MRI scanners. Another example is the wafer handling and the wafer characteristics

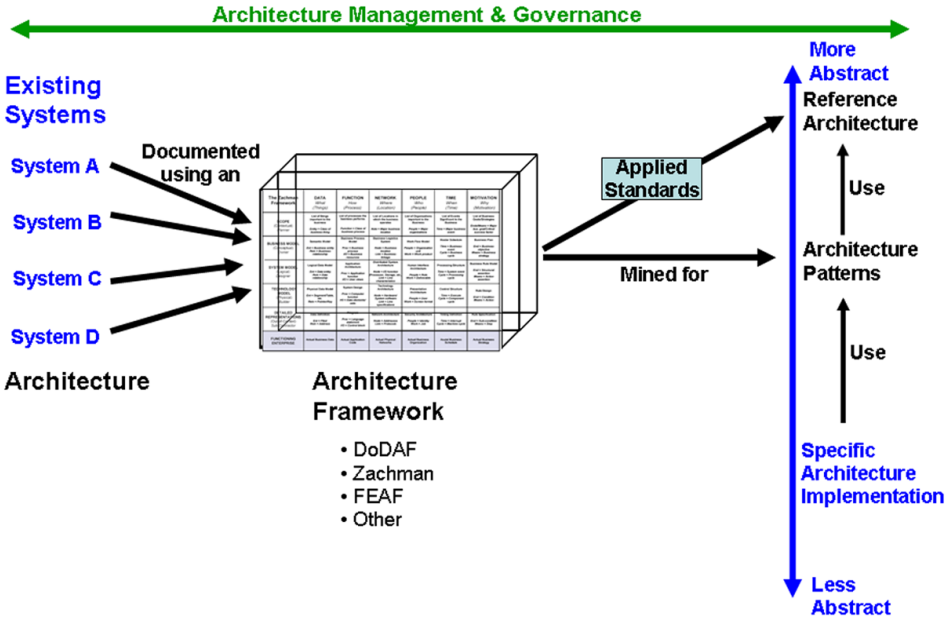


Figure 6. Reference Architecture Development. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

are very concrete and specific but again are reused over generations of wafer scanners.

The Chief Architect (Business plus Technical) owns the Reference Architecture. Ownership is a critical success factor for a Reference Architecture. Sponsorship of business managers for Reference Architectures is a prerequisite. Such sponsorship works only if the Reference Architecture provides value for the business, for example as a decision making tool for business leaders.

Reference Architectures are often related to organization structures. This relationship between organization and architecture is already a heated subject of discussion for actual architectures:

- Should the architecture follow the organization structure, or vice versa?
- Should we compromise architectural integrity to align better with the organization?
- Or, should we adapt the organization to serve the desired architecture?

We have already noted that Reference Architectures often span multiple organizations, complicating this issue even more. The maturity of the organization, the architects, and the architecture is one of the selection criteria. The most mature entity will be normally be leading. Growth over time may result in changes in the leading entity.

The position of Reference Architectures as elaboration of mission, vision and strategy, suggests that Reference Architectures (ought to) play a significant role at enterprise level. This triggers a number of questions as well:

- What policy makers or boards should be involved with Reference Architectures?
- What policy processes and planning tools relate to Reference Architectures?
- Or, in summary, how are Reference Architectures embedded in the enterprise?

5.5. Value of Reference Architectures

Up to this point, the authors have looked at what constitutes a Reference Architecture, and how a Reference Architecture may be applied. However, why is it important to have a Reference Architecture? What is the value of a Reference Architecture when time is required of critical resources, chief architects, for the creation and maintenance of a Reference Architecture? What is the value of Reference Architectures that justify the investment? No actual data on the value of Reference Architectures exist today. If Reference Architectures are not perceived as valuable, then they are an easy target for cost reduction. Currently, there are no indicators to measure the value of Reference Architectures. Nevertheless, the sanity of Reference Architectures requires some kind of feedback loop, which in turn requires some indicators or metrics of Reference Architectures. An example of such a metric may be derived of an experienced benefit in project organizations is faster and better creation of bid proposals. Other identified value propositions for architecture patterns [Cloutier, 2006] also seem applicable to Reference Architectures. They are listed below:

- **Knowledge Management:** Enables reuse of good concepts and implementations, and to preserve them for future projects.
- **Control Complexity:** Architectural patterns may help in controlling the complexity of an architecture by standardizing it on a well-known and practiced pattern.
- **Common Understanding:** Describing parts of the architecture in the context of known and understood architecture patterns results in a common understanding of the architecture.
- **Mitigate Risks:** Using and applying known architecture patterns in architecture design introduce less risk than a new architecture design.

5.5.1. An Industrial View on the Value of the Reference Architecture

The participants in the Systems Architecture Forum identified the following points as justification for using Reference Architectures in industry. Note that these points are consistent with the value propositions from the previous section.

- Reuse and commonality throughout product lines as well as product generation. The potential benefit of this being shorter development cycles and reduced cost from not having to start from scratch.
- Risk reduction is another potential benefit through the use of proven and partly prequalified architectural elements. The general maturity and experience level associated with a Reference Architecture also bears the promise of a higher quality end product.
- Interoperability was highlighted as another major motivation for using Reference Architectures. This aspect is quite different than the reuse aspect. The issue was not using a Reference Architecture as a "template" in which an architect is provided a head-start on a development effort, but, instead, the Reference Architecture was aimed at interoperability to improve compliance for a given context.
- A third aspect that was discussed was the role of a Reference Architecture as a knowledge repository which facilitates knowledge transfer and communication. A Reference Architecture aids the understanding of the basic architectural and design principles. A Reference Architecture can also serve as a framework and lexicon of terms and naming conventions, as well as structural relationships within a company, industry or a domain.
- The final main aspect that was discussed, took an acquisition point of view. An acquisition program backed up by a strong Reference Architecture that ensures interoperability and "form, fit, and function" compatibility promotes flexibility in the choice of suppliers, as well as a lower risk through multisourcing.

6. SUMMARY AND RECOMMENDATIONS

Reference Architectures capture knowledge from existing architectures. Based on an elaboration of mission, vision, strategy, and on customer needs, the Reference Architecture is transformed into an architecture that provides guidance to multiple organizations that evolve or create new architectures.

Reference Architectures should address technical architectures, business architectures, and context. One of the main challenges is to make the inherently abstract Reference Architecture concrete and understandable by providing sufficient specific information and guidelines.

The value of Reference Architectures is foreseen in environments with a high multiplicity factor creating social, organizational, business, application and technical complexity (Figure 7).

With the increased interest in the concept of Reference Architectures, there appears to be a few distinct advantages for a company or organization to identify, document, and use Reference Architectures. One of the advantages is to facilitate reuse, and thereby harvest potential savings through reduced cycle times, cost, risk and increased quality. Another advantage is in the potential for increased interoperability, both within corporations as well as within industries and domains.

Reference Architectures must be actively managed by the organization. The management of the Reference Architectures must ensure that they are updated based on the evolution of the governing standards, emerging technology, changing stakeholder requirements, and maturing business.

Recommended items for further study are:

1. There needs to be a better definition of the required information for a Reference Architecture based on research and comparative analysis of existing Reference Architectures and applications of Reference Architectures. Are the needs for documenting all Reference Architectures created equal? Are the requirements for documenting a space system drastically different than that of a cell phone—or can the Reference Architecture contents be generalized to apply to both?
2. Who is the intended audience of a Reference Architecture? Some suggest that the business is the consumer for strategic planning, and others assert that the consumer of the Reference Architecture is the architect; but they are generally the creators, and as such, know the information implicitly. Finally, others believe the designers are the real users. If it is all three, what is the proper level of detail? Is that different if the user is only the designer?
3. As discussed here, the Reference Architecture seems to be domain specific but this may be dependent on how one defines domain. Reference Architectures like those from Sun and BEA are defined for hardware or web servers, while the Burton group is a generic Reference Architecture for creating Reference Architectures, and yet the Reference Architecture for Space System Data Systems is clearly domain specific.
4. Currently, no quantitative data exists to support the value assertions; they are based on anecdotal evidence. Research should be initiated to prove or disprove the overall value of Reference Architectures to businesses. These observations trigger a set of questions:
   - Is there an end-of-life for Reference Architectures? Can we create a list of criteria to assess the maturity or obsolescence of a Reference Architecture? Does the notion of generation break exist for Reference Architectures? Older Reference Architectures, well embedded and aligned in the organizations, may stifle changes and innovation. Disruptive technologies may kill the value of a Reference Architecture suddenly.
   - How to do change control or management of the Reference Architecture? What is the level of formalization of a Reference Architecture? Which stakeholders are involved in the change management?
5. Crucial to future research is the development of metrics to evaluate Reference Architectures. Reference Architecture reuse may be considered as a competitive edge, a means of reusing or repurposing implicit knowledge made explicit. However, further research needs to be done to quantify the possible competitive edge.

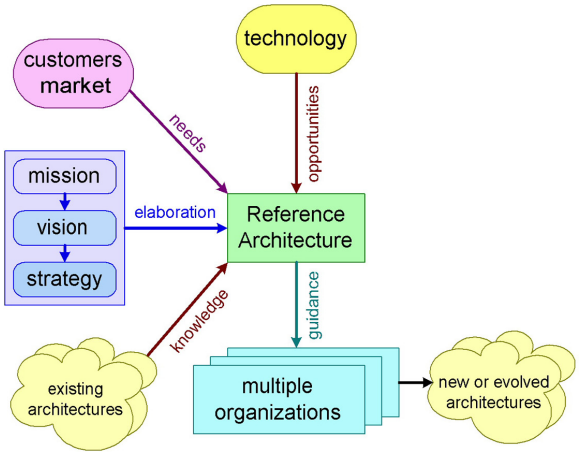


Figure 7. Summary of the role of Reference Architectures, for simplicity feedback loops are not shown. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

REFERENCES

B. Appleton, Patterns and software, essential concepts and terminology, http://www.cmcrossroads.com/bradapp/ docs/patternsintro.html, accessed June 22, 2005.

CCSDS 311.0-R-1, Reference architecture for space data systems, Draft Recommended Practice, Issue 1, Consultative Committee for Space Data Systems, NASA, Houston, TX, January 2007.

R.J. Cloutier, Applicability of patterns to architecting complex systems, Doctoral Dissertation, Stevens Institute of Technology, Hoboken, NJ, 2006.

R.J. Cloutier and D. Verma, Applying pattern concepts to systems (enterprise) architecture, J Enterprise Architecture 2(2) (May 2006): 34–50.

DoDAF, Department of Defense Architecture Framework, Version 1.5, Volume 1, http://jitc.fhu.disa.mil/jitc_dri/pdfs/dodaf_v1v1.pdf, accessed April 23, 2007.

E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design patterns: Elements of reusable object oriented software, Addison-Wesley, Reading, MA, 1995.

R. Hanmer and K. Kocan, Documenting architectures with patterns, Bell Labs Tech J 9(1) (2004), 143–163.

M. Hahsler, Free/open source software development, Koch Stefan (Editor), Idea Group Publishing, 2004, 2005, pp. 103–124.

E. Hole, Architectures as a framework for effective and efficient product development in a dynamic business environment, Proc Third Annu Conf Syst Eng Res, March 2005.

INCOSE Systems Engineering Handbook 2a, Seattle, WA, 2000.

IEEE Std. 1471-2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, IEEE Std. 1471-2000, IEEE, New York, 2000.

IEEE Std. 1220-2005, IEEE Standard for Application and Management of the Systems Engineering Process, IEEE Std. 1220-2005, IEEE, New York, 2005.

ISO/IEC 10746-1, RM-ODP ISO Reference Model for Open Distributed Processing (RM-ODP), 2004.

A. Martelli and D. Ascher, Python cookbook, O'Reilly, 2005.

R. Martin, Reference architecture, Uncle Bob's software craftsmanship corner, October 2003 http://www.artima.com/weblogs/viewpost.jsp?thread=17799

D. McIlroy, Mass produced software components, Proc 1968 NATO Conf Software Eng, October 1968, http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF

G. Muller, CAFCR: A multi-view method for embedded systems architecting; balancing genericity and specificity, Ph.D. thesis, June 2004, retrieved from http://www.gaudisite.nl/ThesisBook.pdf, Chapter 14.

M. Maier and E. Rechtin, The art of systems architecting, 2nd edition, CRC Press, Boca Raton, FL, 2000.

E. Rechtin, The art of systems architecting, Prentice Hall PTR, Upper Saddle River, NJ, 1991.

P. Reed, Reference architecture: The best of best practices, IBM developer works, September 2002, http://www-128.ibm.com/developerworks/rational/library/2774.html

M. Rosen, Enterprise architecture trends 2007: The year ahead, Cutter Executive Report, September 2002, http://www.cutter.com/offers/EAtrends.html

S. Simmons, Introducing the WebSphere integration reference architecture, IBM WebSphere Dev Tech J (August 2005), http://www-128.ibm.com/developerworks/ websphere/techjournal/0508_simmons/0508_simmons.html

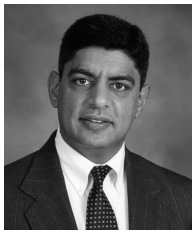
R. Sanz and J. Zalewski, Pattern-based control systems engineering, IEEE Control Syst Mag (June 2003), 43–60.

State of Arizona Target Security Architecture Information Technology (IT) Technical Document, Revision 2.0, dated April 5, 2004. Downloaded from http://www.azgita.gov/ enterprise_architecture/NEW/Security_Arch/ AZ_Target_Security_Architecture.htm

J. Zachman, A framework for information systems architecture, IBM Syst J 26(3) (1987), 276–290.

Rob Cloutier is an Associate Professor of systems engineering in the School of Systems and Enterprises at Stevens Institute of Technology. He has over 20 years experience in systems engineering & architecting, software engineering, and project management in both commercial and defense industries. Industry roles included lead avionics engineer, chief enterprise architect, lead software engineer, and system architect on a number of efforts and proposals. His research interests include model-based systems engineering and systems architecting using UML/SysML, reference architectures, systems engineering patterns, and architecture management. Rob holds a B.S. from the US Naval Academy, an MBA from Eastern College, and his Ph.D. in Systems Engineering from Stevens Institute of Technology.

Gerrit Muller received his Master's degree in physics from the University of Amsterdam in 1979. He worked from 1980 until 1997 at Philips Medical Systems as a system architect, followed by two years at ASML as a manager of systems engineering, returning to Philips (Research) in 1999. Since 2003 he has worked as a senior research fellow at the Embedded Systems Institute in Eindhoven, focusing on developing system architecture methods and the education of new system architects, receiving his doctorate in 2004. In January 2008 he accepted a full professor of systems engineering appointment at Buskerud University College in Kongsberg, Norway.



Dinesh Verma received the Ph.D. and the M.S. in Industrial and Systems Engineering from Virginia Tech. He is currently serving as Dean of the School of Systems and Enterprises and Professor in Systems Engineering at Stevens Institute of Technology. During his six years at Stevens he as successfully proposed research and academic programs exceeding $50m in value. Verma concurrently serves as Scientific Advisor to the Director of the Embedded Systems Institute in Eindhoven, Holland. Prior to this role, he served as Technical Director at Lockheed Martin Undersea Systems, in Manassas, Virginia, in the area of adapted systems and supportability engineering processes, methods and tools for complex system development and integration. Verma continues to serve numerous companies in a consulting capacity, to include Eastman Kodak, Lockheed Martin Corporation, L3 Communications, United Defense, Raytheon, IBM Corporation, Sun Microsystems, SAIC, VOLVO Car Corporation (Sweden), NOKIA (Finland), RAMSE (Finland), TU Delft (Holland), Johnson Controls, Ericsson-SAAB Avionics (Sweden), Varian Medical Systems (Finland), and Motorola. Dr. Verma has authored over 100 technical papers, book reviews, technical monographs, and co-authored two textbooks. He is a Fellow of the International Council on Systems Engineering (INCOSE), a senior member of SOLE, and was elected to Sigma Xi, the honorary research society of America. He serves as on the Core Curriculum Committee of the Delft University's Space Systems Engineering Program (Holland) and as an advisor to the Systems Engineering Center of Expertise at the Buskerud University College (Norway). He was honored with an Honorary Doctorate Degree (*Honoris Causa*) in Technology and Design from Växjö University (Sweden) in January 2007.

Dr. Roshanak Nilchiani is an Assistant Professor at the School of Systems and Enterprises at Stevens Institute of Technology. She is currently working on physical and mathematical modeling of systems' response to change. Her research interests include Risk-Based Complex Engineering Systems Design and Operations, Modeling and Assessing the Resilience of Infrastructure Systems, System of Systems Design, and Agile Systems and Enterprises. During her time at MIT, Dr. Nilchiani performed research on flexible designs for DARPA's Orbital Express system, direct broadcasting satellites, and next generation Mars rovers. In addition she has served as a mission analysis and design consultant for 4Frontiers, a commercial space travel company. Dr. Nilchiani received her Ph.D. in Aerospace Systems from the Massachusetts Institute of Technology in 2005 and is an associate member of the New York Academy of Sciences, and a member of the American Institute of Aeronautics and Astronautics.

Eirik Hole is a Lecturer in Systems Engineering and Engineering Management in the School f Systems and Enterprises at Stevens Institute of Technology. Mr. Hole obtained his MSc (Dipl. Ing.) in Aerospace Engineering from the University of Stuttgart, Germany in 1995. Before coming to Stevens in 2003, Mr. Hole spent 8 years in various systems engineering related positions. He started his career on the systems engineering team for a new generation antiship missile system. Here he was involved in system specification, overall system design, and the planning of major design reviews. He was also engaged in systems engineering process improvement and the implementation of a systems engineering tool set. Before coming to Stevens, Mr. Hole worked as a consultant in the area of applied systems engineering and requirements management as well as on the practical application and implementation of systems engineering and requirements management tools. His consulting experience includes customers from automotive, defense, and consumer goods industry in Europe.

Mary Alice Bone has worked as a Systems Engineer for GE Transportation, BAE Systems, and Rockwell Collins. She holds a B.S. in Aerospace Engineering from the University of Missouri—Rolla and a M. Eng. in Systems Engineering from Iowa State University. She has been involved with internal company system process groups which has ignited the interest in producing system requirement processes that resolve the struggle between system process and project process while maintaining systems engineering integrity. She is currently pursuing a Ph.D. in Systems Engineering from Stevens Institute of Technology.