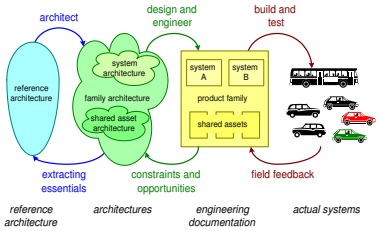
A Reference Architecture Primer

-



Gerrit Muller
University of Southeast Norway-NISE
Hasbergsvei 36 P.O. Box 235, NO-3603 Kongsberg Norway
gaudisite@gmail.com

Abstract

A Reference Architecture captures the essence of the architecture of a collection of systems. The purpose of a Reference Architecture is to provide guidance for the development of architectures for new versions of the system or extended systems and product families.
We provide guidelines for the content of a Reference Architecture and the process to create and maintain it. A Reference Architecture is created by capturing the essentials of existing architectures and by taking into account future needs and opportunities, ranging from specific technologies, to patterns to business models and market segments.

All Gaudí documents are available at:
http://www.gaudisite.nl/

version: 0.6 status: preliminary draft June 5, 2018

1 Introduction

We will start with discussing *why*, *what*, *when* and *how* of Reference Architectures in Section 2. One of the main challenges of creating and using Reference Architectures effectively is its level of abstraction: How much detail should be included? We elaborate this aspect in Section 3. In Section 4 we provide some insights on what should be in the Reference Architecture.

2 Reference Architectures in general

The text in this section is partially borrowed from [5]. We discuss *why*, *what*, *when*, and *how* of Reference Architectures in this section.

2.1 Why Reference Architectures?

The magic *multi* word.

In all domains we see two simultaneous trends:

- 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, in a highly competitive market, for example with cost and performance pressure.

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 open development at multiple locations (*multi-site*), by multiple vendors, across multiple organizations.

In Figure 1 we also added multi-*, because the multiplicity is not limited to organizations, vendors and locations. Systems also become more multi-domain (e.g. security has military as well as civil applications), multi-application (e.g. electron microscopes are used for metrology in high volume applications and for material analysis in low volume applications), multi-cultural (global application, but customized for local cultural aspects), development and manufacturing is based more often on multi-sourcing, 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, for example by a domain model
- a common (architectural) vision

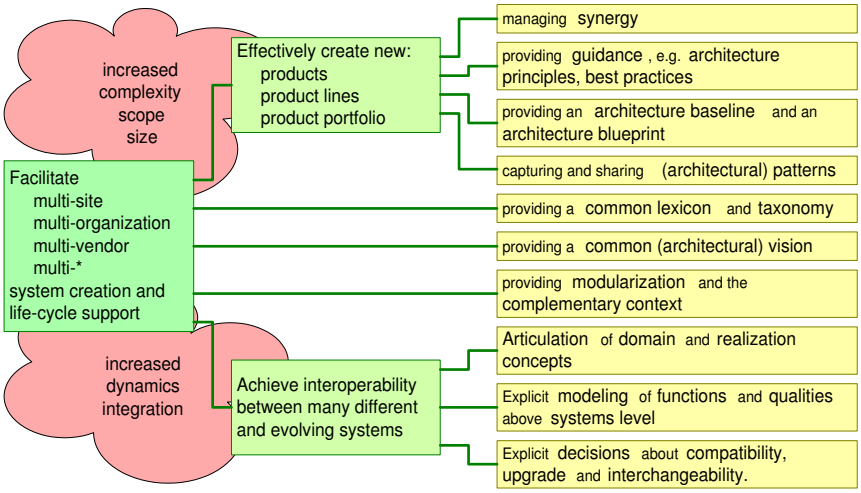


Figure 1: Graph of objectives of Reference Architectures

- 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.

When a shared set of design or implementation assets is used, for example a common set of HW and SW components, then a Reference Architecture facilitates rapid product instantiation by providing the value mentioned above.

Effective creation of products, products lines, and product portfolios

In this setting the goal is to effectively create products, products lines, and product portfolios. The Reference Architecture improves the effectiveness by:

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

Driving and harvesting 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 at the 1968 NATO conference about Software Engineering [1].

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 re-occurrence 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, where is described in 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 blueprint for future architectures. Again this hopefully prevents the re-invention and re-validation of solutions for already solved problems. This baseline is the starting point to support the required variation.

Achieving interoperability between many different and evolving systems

In this multi-* world interoperability determines the usability, performance and dependability of user level applications. Reference Architectures must improve interoperability by:

- Articulation of domain and realization concepts.
- Explicit modeling of functions and qualities at context level, going beyond individual system 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 re-use to be effective it is required that integration effort must be small.

2.2 When to Use Reference Architectures?

Figure 2 shows in a different way than Figure 1 that Reference Architectures start to have value when the multi-* factor is large enough. When creating a single system, we need engineering, design and architecting competencies. However, when the scope increases and multiple product creations are coupled, then Reference architectures are indicated. For small stand-alone developments Reference Architectures are overkill.

An explicit Reference Architecture facilitates evolution of a product portfolio by providing explicit insight from market to realization. However, evolution will only happen more smooth if the Reference Architecture is not used as the *holy* reference. What has been documented can be changed and should not be viewed as unmutable.

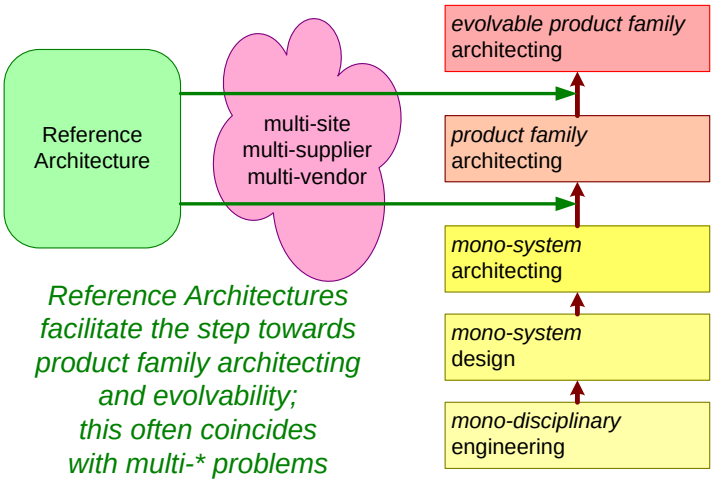


Figure 2: When to Use Reference Architectures

2.3 What do Reference Architectures contain?

A Reference Architecture is strongly linked to company (or consortium, e.g. MIPI) mission, vision and strategy, see Figure 3. Note that mission, vision and strategy are relatively stable entities, with a history (experience) and a future (needs and potential changes). The strategy determines what multi-dimensions 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.

A Reference Architecture facilitates a shared understanding across multiple products, organizations, and disciplines about current architecture(s) and future

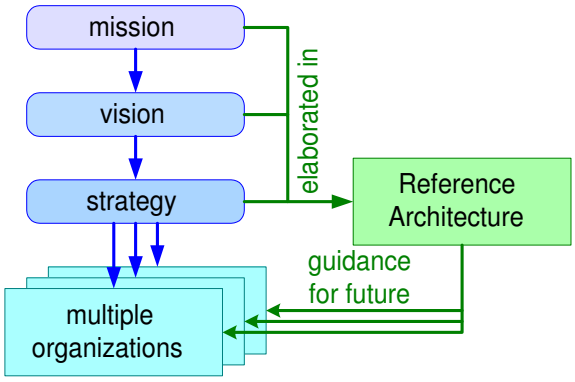


Figure 3: Reference Architecture elaborates Mission, Vision and Strategy

directions.

Architectures of the past are transformed in a Reference Architecture. However, the purpose of the Reference Architecture is future oriented. The mission, vision and strategy are needed to add the future direction to the wisdom of the past. Note that future directions are inherently unproven. Hence future directions might be conflicting with the experience as will be further discussed in Subsection 2.5 that reference architectures should only contain proven concepts.

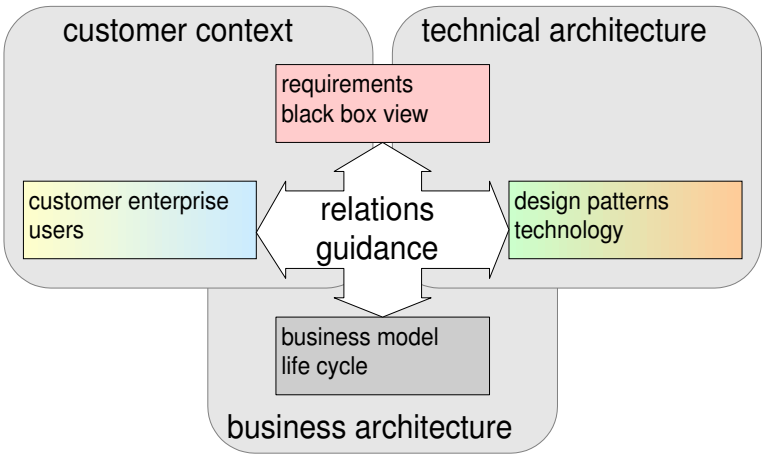


Figure 4: Reference Architecture = Business Architecture + Technical Architecture + Customer Context

Figure 4 shows that a Reference Architecture should address:

- Technical architecture,
- Business architecture, and
- Customer context.

In practice, business architecture and customer context are often missing, see [8]. As a consequence these technical reference architectures represent solutions for unspecified problems in unspecified contexts.

Figure 4 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. 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 Reference

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

2.4 How to Use Reference Architectures?

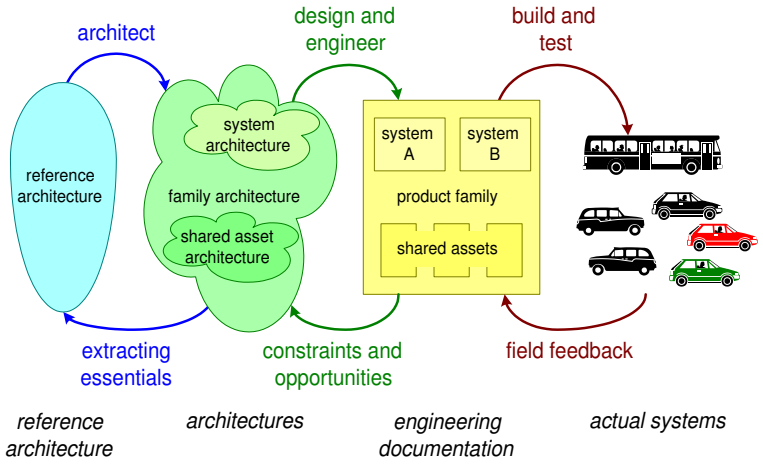


Figure 5: Instantiation of a Reference Architecture in few transformations

The level of abstraction of Reference Architectures makes it more difficult to understand their role. Figure 5 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. 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.

The re-use 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 (also often called platform).

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.

2.5 What are inputs of a Reference Architecture?

A Reference Architecture captures previous experience, for instance by mining, or by generalizing existing architectures. 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.

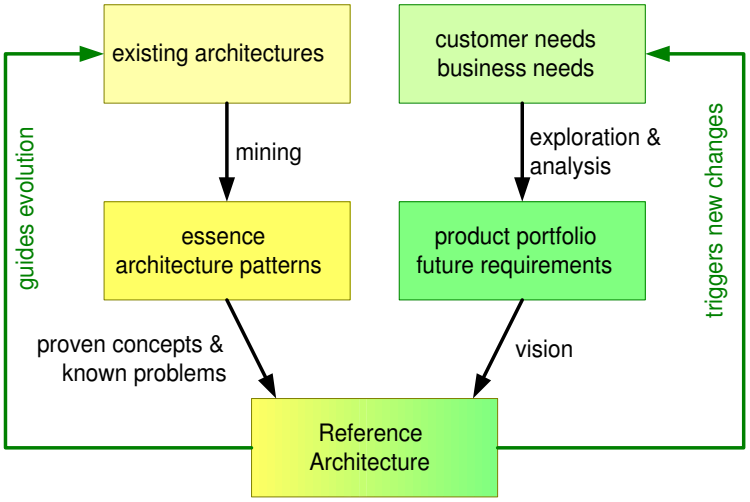


Figure 6: Inputs of a Reference Architecture

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 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.

2.6 Criteria for a good Reference Architecture

We recommend the following list as criteria for a good Reference Architecture:

- understandable for a broad set of heterogeneous stakeholders (customers, product managers, project managers, engineers et cetera).
- accessible and actually read/seen by majority of the organization
- addresses the key issues of the specific domain
- satisfactory quality
- acceptable
- up-to-date and maintainable
- adds value to the business

The understandability is crucial for all the goal formulated in Subsection 2.1. The challenge is to make it understandable for the wide variety of stakeholders. The proof of the pudding for Reference Architectures is the amount of people in the organization that have actually seen and read the Reference Architecture. Note that security concerns sometimes conflict with the necessity for information sharing and open communication. A *secret* Reference Architecture will not work.

The quality level is assessed by the stakeholders and by historical evaluation. One of the threats to quality is the acceptance. Sometimes quality of the architecture itself or of the description is sacrificed in favor of political arguments (acceptance). Unfortunately, a Reference Architecture will only survive a limited set of compromises. A heavily compromised Reference Architecture potentially threatens the survival of all derived products.

An outdated Reference Architecture hampers the business and loses its credibility in the organization. To be up-to-date and maintainable is related to the level of abstraction of the Reference Architecture, discussed in Section 3.

The bottom-line criterium for a Reference Architecture is the value to the business. If the Reference Architecture does not add value to the business, then why bother?

3 Level of abstraction

One of the most crucial questions when creating a Reference Architecture is: what is the appropriate level of abstraction? In Section 2, specifically in Figure 5, we have shown that a Reference Architecture is inherently abstract. There is a severe danger of being over-abstract. When the description is over-abstract then many stakeholders will ignore it, nullifying the value of the Reference Architecture. The

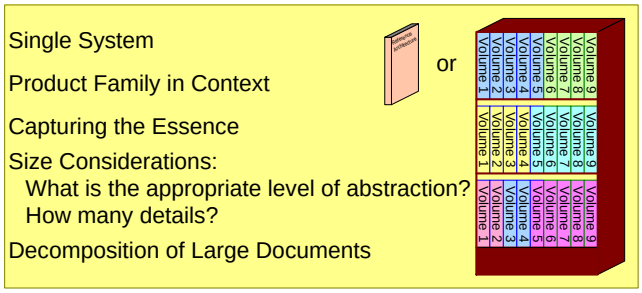


Figure 7: Challenge: Appropriate Level of Abstraction

opposite danger is that so much practical elaboration is provided that both the effort of creation is well as of using the Reference Architecture is too big. In Figure 7 those two extremes are visualized as single small document or a cabinet full of books. The Figure also shows the subjects we will discuss in the remainder of this section.

3.1 Number of details in a single system

The translation of system requirements of one specific system into detailed mono-disciplinary design decisions spans many orders of magnitude. The few statements of performance, cost and size in the system requirements specification ultimately result in millions of details in the technical product description: million(s) of lines of code, connections, and parts. The technical product description is the accumulation of *mono-disciplinary* formalizations. Figure 8 shows this dynamic range as a pyramid with the system at the top and the millions of technical details at the bottom.

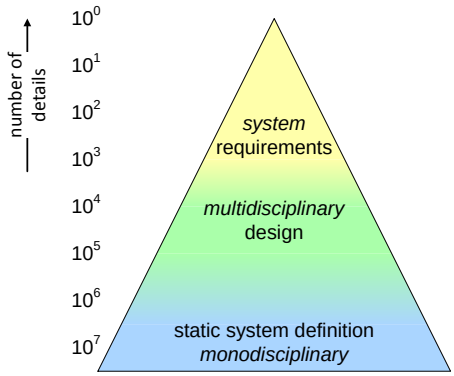


Figure 8: Level of Abstraction Single System

3.2 From single system to product family in context

Reference Architectures address multiple products in one or more product families. As discussed in Section 2 the Reference Architecture also has to address the context of the system, both from customer as well as business perspective. We can transform Figure 8 with the number of details of a single system into Figure 9 to show the number of details of a product family in its context. Note that the number of details of the product family is represented by an increased pyramid, due to the increased scope. The context is shown also as a pyramid, representing the fact that in the outside world where systems are actually used also can be viewed at many levels of abstractions.

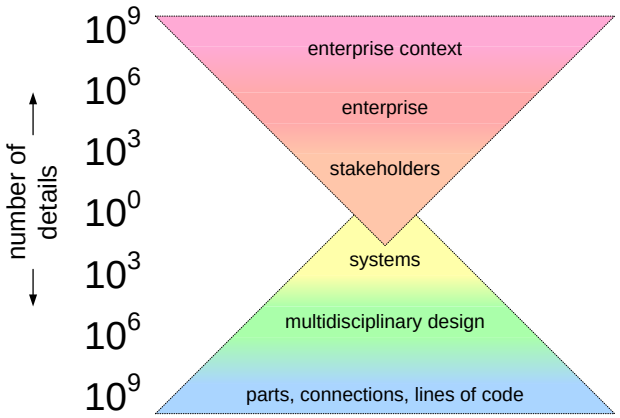


Figure 9: Product Family in Context

3.3 Reference Architecture coverage

The challenge of developing a Reference Architecture is to capture the essence of both the systems to be build as well as the contexts where systems are being used. Figure 10 shows that most of the Reference Architecture covers the higher abstraction levels, however some crucial details either from mono-disciplinary area or from the customer or business contexts might have to be included.

3.4 What is an appropriate size of a Reference Architecture?

We can rephrase the question of the appropriate level of abstraction into the question how much information (specific facts) should be included in the Reference Architecture. Figure 11 shows a spectrum of possibilities for Reference Architecture descriptions on a logarithmic axis representing the number of details or specific facts that are included. Two examples are provided in this spectrum:

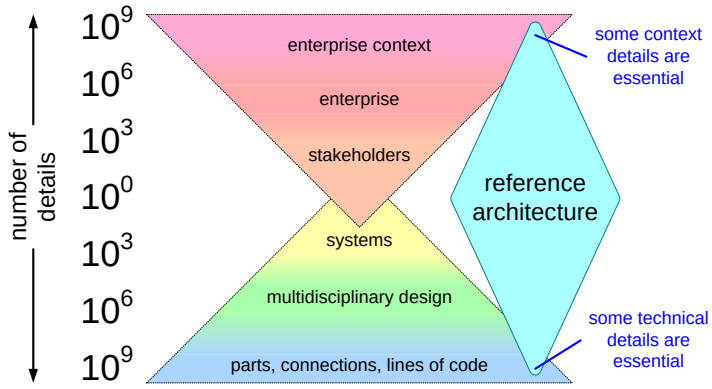


Figure 10: RA: Capturing the Essence

high level, 1k details this Reference Architecture, for example, contains 7 subjects:

- Market description and customer key drivers; capturing the customer environment
- Process flow; capturing how the customer operates
- Key performance parameters; capturing the key performance of the system
- Decomposition and information model; capturing the main concepts of modularization and information
- Concurrency and synchronization; capturing the more detailed design of concurrency within the system

elaborated, 1M details this Reference Architectures is much more elaborated containing 29 subjects:

- 6 views to capture the customer objectives and environment
- 3 views to capture the customers operation
- 5 views capturing features and functions of the systems
- 9 conceptual views to capture the technical architecture
- 6 more detailed views of the technical architecture, zooming in on key design decisions

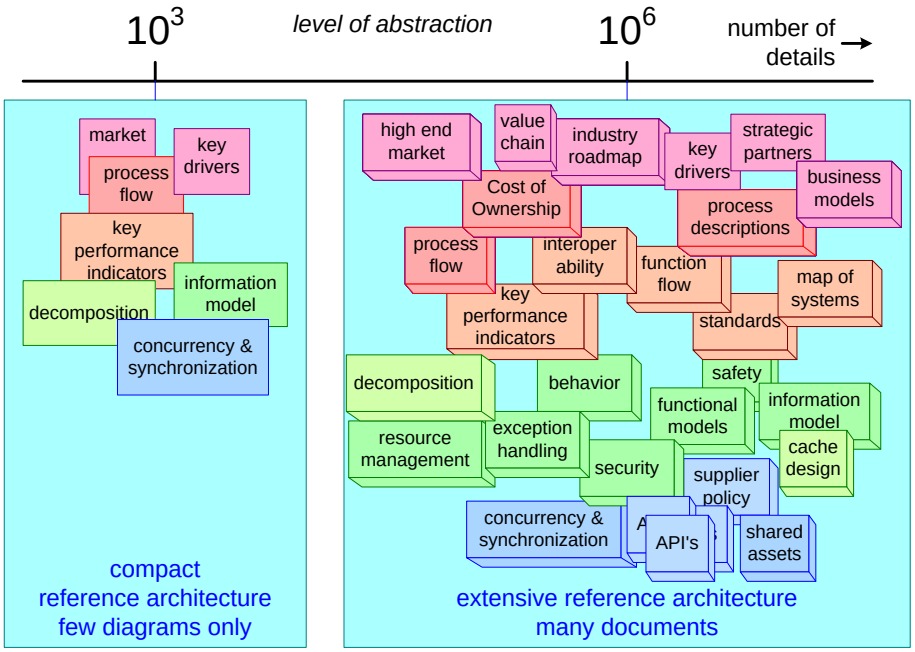


Figure 11: RA: level of abstraction, number of details

Figure 12 annotates the spectrum of Figure 11 with characteristics and size considerations. The *high level* Reference Architecture requires a (relatively[1]) low effort to create, maintain and read the Reference Architecture. The down side is that it provides only limited guidance and anchor value.

The *elaborated* Reference Architecture requires a lot of work to create, worse a lot of work is continuously required to maintain it. Also reading such an elaborated version is much more work. Most stakeholders will never be able to read the entire description, because they don't have the time to do so. However, the elaborated parts will provide much more guidance locally. The concurrency and synchronization document, for example, might provide a ready-to-go prescription.

3.5 Designing the Reference Architecture description

The description of a Reference Architecture deserves design attention. All design principles for systems, such as decomposition, coupling and cohesion, also hold for documents. Figure 13 shows a summary of how to cope with granularity of documents as described in [2]. The main messages in this document are:

[1] Writing a relevant compact document is a challenging activity. Quite some time and iterations might be needed to reduce the description to the essential minimum. It is much more easy to create a large stack of data.

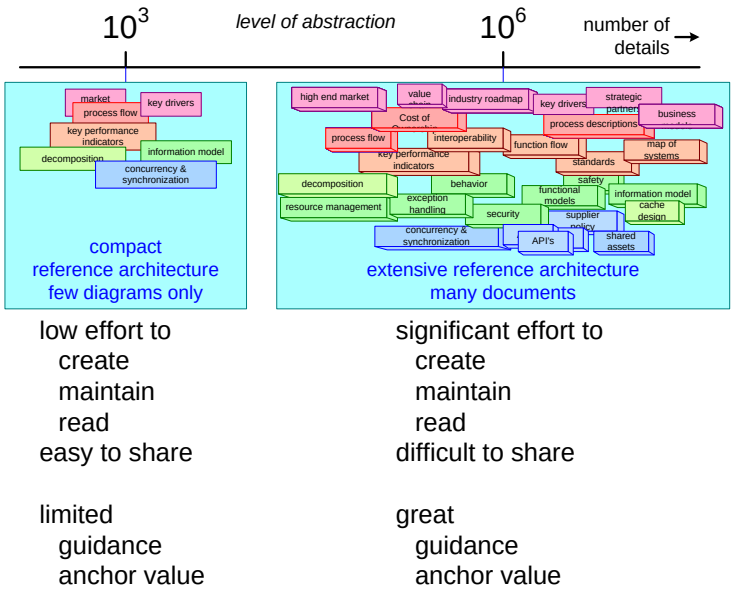


Figure 12: Size Considerations

- Large documents should be decomposed in smaller document.
- When decomposing the structure should be captured and an overview should be provided as service to the stakeholders.
- *Atomic* documents, i.e. documents that are themselves not further decomposed, should be small enough to create, maintain and read in limited time.
- Pragmatic guidance for size is maximum 20 pages per document (including meta-information), preferably about 10 pages.
- The content should be a mixture of diagrams, tables and explanatory text.
- Decomposition can be based on stakeholders, concerns, authorship and time of availability.

4 What should be in Reference Architectures?

The core of the matter is: what should be present in a Reference Architecture? We will discuss the following aspects:

- Guidance from Best Practices
- Visualizations

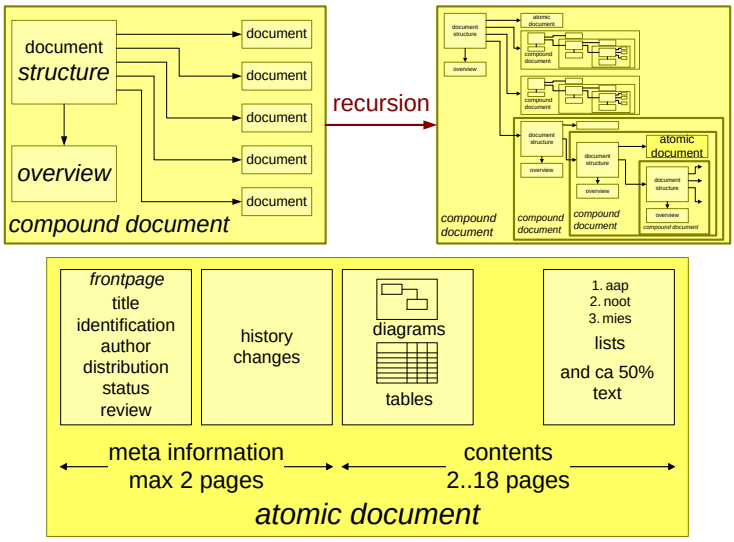


Figure 13: Decomposition of Large Documents

- Structure
- What content should be in Reference Architectures?

4.1 Guidance from Best Practices

The System Architecting Forum (`www.architectingforum.org`) is a meeting of experienced architects from multiple domains. In every meeting one or two subjects are discussed. During these discussions *best practices* are identified and published at the web-site. A number of the best practices from the first meetings provide guidance for the content of Reference Architectures:

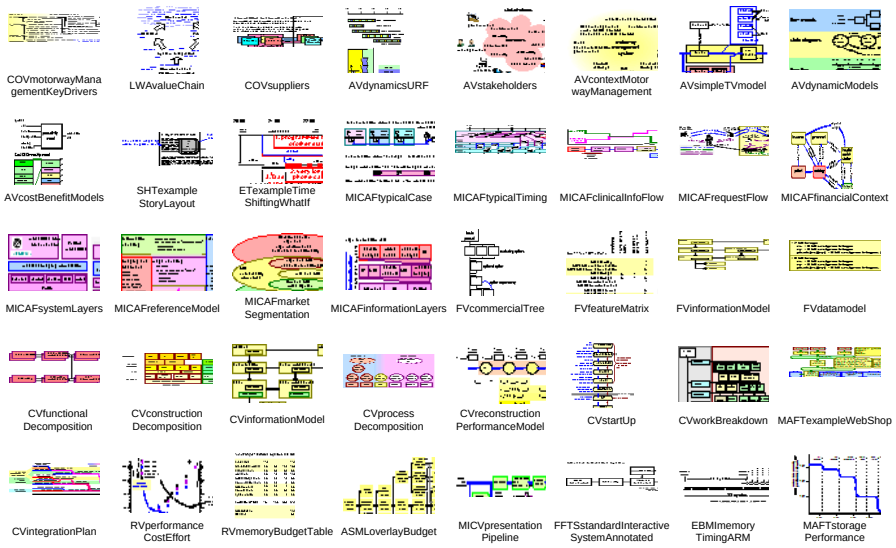
- 1.1 [6]. One of several prerequisites for architecture creative synthesis is the definition of **5-7 specific key drivers** that are critical for success, along with the rationale behind the selection of these items.
- 2.1 [4]. he essence of a system can be captured in about **10 models/views**.
- 2.2 [4]. A **diversity** of architecture descriptions and models is needed: languages, schemata and the degree of formalism.
- 2.3 [4]. The level of **formality** increases as we move closer to the implementation level.

We recommend that a Reference Architecture explicitly describes 5-7 specific key drivers and the rationale behind the selection of these items. The key drivers

are often the business objectives or the objectives of the main stakeholders such as the customer. Further, based on the same best practices we recommend to capture about 10 different models or views. Many more models or views are needed to describe a system. However, in practice about 10 models or views are perceived to be manageable number. This is reflected by the fact that in practice about 10 views or models dominate the description and the discussions.

We will elaborate the diversity of descriptions somewhat more in Subsection 4.2. Finally the best practice about formality in combination with the positioning of Reference Architectures in Figure 10 tells us that Reference Architectures are far removed from implementation details. The description of Reference Architectures has a low degree of formality. Rather, the description of Reference Architectures must be accessible for a broad an rather heterogeneous group of stakeholders, as discussed in Subsection 2.6.

4.2 Visualizations



actual figures and references to their use at http://www.gaudisite.nl/figures/<name>.html

Figure 14: Possible useful visualizations

Reference Architectures deal with a very broad and heterogeneous set of issues, as shown in Figure 10. The description of a Reference Architecture will have to borrow appropriate languages and schemata from the many involved disciplines. Considering the fact that we deal with tens of disciplines, where every discipline uses tens of schemata and languages, we can borrow from hundreds of schemata and languages!

The Gaudí site (`www.gaudisite.nl`) provides inspiration for useful visualizations. A subset is shown in Figure 14.

4.3 Structure

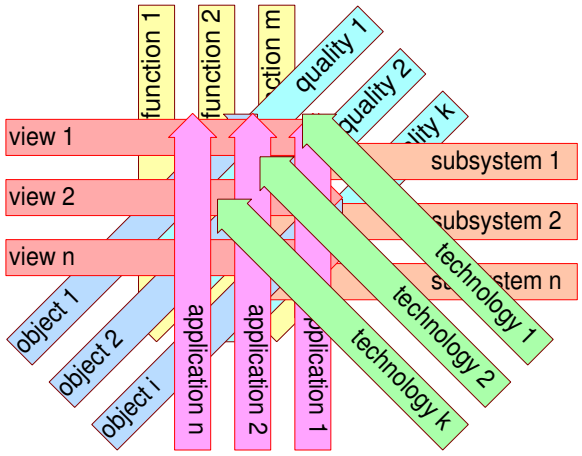


Figure 15: Ideal structure for Reference Architectures does not exist

There are many possible dimensions that can be used to structure the Reference Architecture. Unfortunately no single dimension is ideal to structure. The structure of the Reference Architecture must serve its communication purpose. In other words the structure itself is less important than clarity and understandability of the content.

4.4 What content should be in Reference Architectures?

The main focus of most people when creating architecture documentation is on *decomposition* and *interfaces*. Both aspects are important and a Reference Architecture should provide guidance for both aspects. However, besides decompositions and interfaces guidance and insights must be communicated that are synthesis oriented: How do the components fit together to create a well-performing system? Figure 16 gives an example for the technical architecture part of the Reference Architecture.

The technical architecture typically has several decompositions, such as the decomposition in building blocks as present in the repositories, and the functional decomposition. To work with multiple decompositions we need an explanation of the relations between these decompositions, for example by providing an allocation: what are the building blocks that contribute to a certain function? The synthesis and integration, however, require many additional views on the technical architecture,

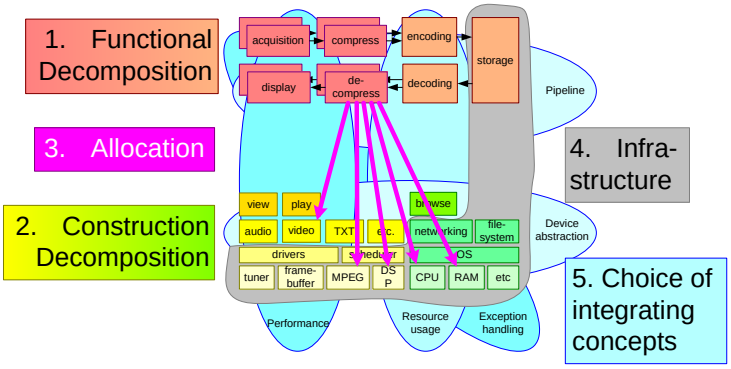


Figure 16: Synthesis, Integration, Relation oriented

for example for aspects such as performance, resource usage, exception handling or device abstraction.

In general the Reference Architecture makes clear how the most relevant qualities are realized and how the most critical design aspects are realized. Qualities and aspects are so-called cross-cutting: the quality is a result of the simultaneous interaction of many building blocks and/or functions. Qualities and aspects are dimensions of description that don't follow the conventional decomposition axis.

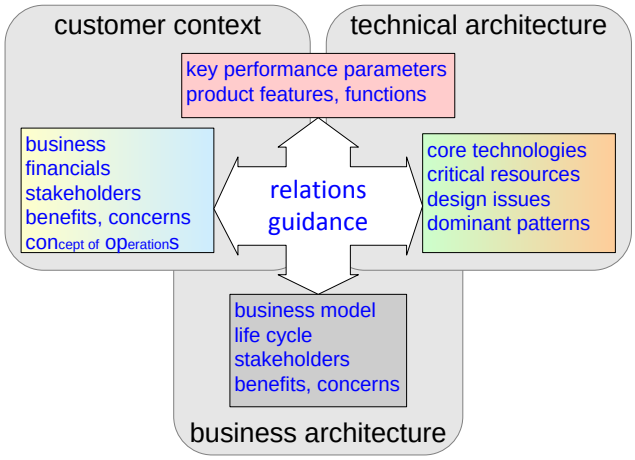


Figure 17: Checklist for Reference Architecture content

So far we have discussed mostly the content of the technical architecture as an example. Figure 17 provides a short checklist for the different parts of the Reference Architecture.

In the Systems Engineering world the *Concepts of Operations* is a well known

document. Jack Ring [7] describes the ConOps as follows:

> A ConOps describes how a community intends to use a contemplated system as a means to mitigate or suppress an actual or anticipated problem situation. A ConOps serves to converge multiple stakeholders toward a common image and understanding of the requested system.
>
> The viewpoint of a ConOps is from the outside-in. A ConOps describes both the stimuli to which the intended system is expected to respond and the effect the responses are intended to have on the situation. Because it describes a system of the future, a system yet to be designed, it is necessarily speculative - a vision – though hopefully not an hallucination. It tells a story, reflects out-of-the-box thinking and is not concerned with immediate perceptions of feasibility.
>
> A ConOps avoids assumptions about the internal content and structure of the eventual system. This is done to avoid getting lost in detail, avoid premature feasibility (mis)judgements and preclude the early insertion of pet design concepts. Such avoidance is demonstrated in this ConOps by placing all observations about possible content and structure in appendices for consideration by designers but not as part of the ConOps baseline.

5 Summary

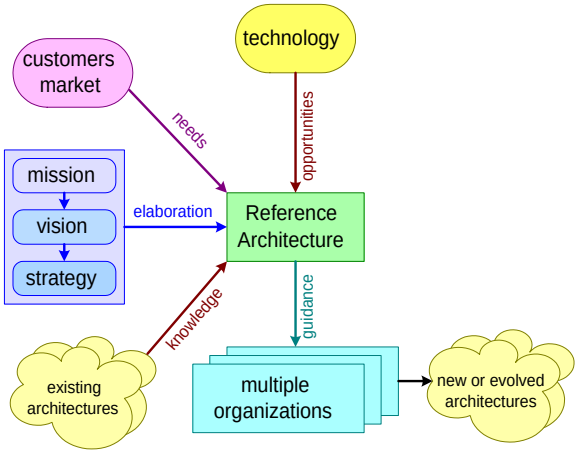


Figure 18: Summary of the role of Reference Architectures

Reference Architectures capture knowledge from existing architectures. Based on an elaboration of mission, vision and strategy, and on future customer needs the

Reference Architecture is transformed in an architecture that provides guidance to multiple organizations that evolve or create new architectures, see Figure 18.

Reference Architectures should address Technical and Business Architectures and the context. One of the main challenges is to make this 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. This is a young area, where more questions are available than answers, ranging from proven value to life-cycle of Reference Architectures.

6 Acknowledgements

The contribution of the members of the System Architecting Forum (`www.architectingforum.org`) in the spring meeting of 2007 provided a good starting point for this paper, see the resulting white paper [5].

The discussions in the Darwin research project (`www.esi.nl/darwin/`) also provided inputs. Piërre van der Laar reviewed the paper.

References

[1] Doug McIlroy. Mass produced software components. In *proceedings of 1968 NATO Conference on Software Engineering*, 1968.

[2] Gerrit Muller. Granularity of documentation. `http://www.gaudisite.nl/DocumentationGranularityPaper.pdf`, 1999.

[3] Gerrit Muller. The system architecture homepage. `http://www.gaudisite.nl/index.html`, 1999.

[4] Gerrit Muller and Eirik Hole. Architectural descriptions and models. `http://www.architectingforum.org/whitepapers/SAF_WhitePaper_2006_2.pdf`. White Paper Resulting from Architecture Forum Meeting March 21-22, 2006 (Washington DC, USA).

[5] Gerrit Muller and Eirik Hole. Reference architectures; why, what and how. `http://www.architectingforum.org/whitepapers/SAF_WhitePaper_2007_4.pdf`. White Paper Resulting from Architecture Forum Meeting March 12-13, 2007 (Hoboken NJ, USA).

[6] Gerrit Muller and Eirik Hole. The state-of-practice of systems architecting: Where are we heading? `http://www.architectingforum.org/whitepapers/SAF_WhitePaper_2005_1.pdf`. White Paper

Resulting from Architecture Forum Meeting October 4-5, 2005 (Helsinki, Finland).

[7] Jack Ring and Wayne Wymore. Concept of operations (conops) of a systems engineering education community (seec). http://www.incose.org/ProductsPubs/products/conops.aspx, 2004. prepared by Education Measurement Working Group, International Council on Systems Engineering (INCOSE).

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

History

Version: 0.6, date: August 20, 2007 changed by: Gerrit Muller

- some small textual changes and additions

Version: 0.5, date: August 13, 2007 changed by: Gerrit Muller
- many small textual changes
- added a quote about ConOps
- added references to SAF white papers

Version: 0.4, date: August 10, 2007 changed by: Gerrit Muller
- created text version
- changed status to preliminary draft

Version: 0.3, date: August 8, 2007 changed by: Gerrit Muller
- added OHTstructure, LWAarchitectureHow
- added content guidance
- added content slides

Version: 0.2, date: July 20, 2007 changed by: Gerrit Muller
- added view/visualization collection

Version: 0.1, date: July 19, 2007 changed by: Gerrit Muller
- added criteria
- defined logo
- added level of abstraction
- compacted document design

Version: 0, date: July 18, 2007 changed by: Gerrit Muller
- Created, no changelog yet