A Classification of Software Reference Architectures: Analyzing Their Success and Effectiveness

Samuil Angelov
School of Industrial Eng.
Eindhoven Univ. of Technology
s.angelov@tue.nl

Paul Grefen
School of Industrial Eng.
Eindhoven Univ. of Technology
p.w.p.j.grefen@tue.nl

Danny Greefhorst
ArchiXL
dgreefhorst@archixl.nl

Abstract

A software reference architecture is a generic architecture for a class of information systems that is used as a foundation for the design of concrete architectures from this class. We observe that certain reference architectures have become more successful than others. One of the reasons for this is the level of congruence between their goals, context, and design. In this paper, we provide a framework for the classification of reference architectures. Using our framework on a set of reference architectures, and based on experiences with reference architectures, we define five main types of reference architectures that have congruent goals, context, and design. Reference architectures that can be classified in one of these types have better chances to become a success. We illustrate our conclusions with a number of reference architectures. This research facilitates design of more effective reference architectures.

1. Introduction

Software reference architectures have emerged as abstractions of concrete architectures from a certain domain. Reference architectures (RA) can be used as an inspiration in the design of concrete architectures or as a standardization tool that guarantees the interoperability between systems and between components of systems [21]. A reference architecture is used for the design of concrete architectures in multiple contexts, affecting different stakeholders in each context [4]. Nowadays, the increasing complexity of software, the need for efficient and effective software design processes and for high levels of system interoperability lead to an increasing role of reference architectures in the software design process. A commonly accepted definition for software reference architectures does not exist. In this paper, we use the definition for a reference architecture provided in [6], according to which a reference model is *"a division of functionality together with data flow between the pieces"*, and a reference architecture is *"a reference model mapped onto software elements (that cooperatively implement the functionality defined in the reference model) and the data flows between them"*. In this paper, e use the term "reference architecture" to refer to the documented description of a reference architecture.

Concrete software architectures are designed on the basis of required functionalities and system, business, and architecture qualities defined by the stakeholders [6]. These functionalities and qualities reflect a specific context and the business goals of the stakeholders. The types of possible goals, the identification of required functionalities and qualities, and their effects on the architecture design have been well-studied and extensively published (see, e.g., [6]). The usage in system design of a well-designed concrete architecture is in a sense guaranteed as the architecture is designed specifically for the development of that system.

The business goals, required functionalities and quality attributes are also considered as the aspects influencing the design and specification of reference architectures [7]. The definition of a reference architecture, however, is not a direct response to a need for a system. Rather, it is an *estimation* that the existence of a reference architecture will facilitate the work of a design team in multiple projects or, even more, the work of multiple design teams in a specific domain. The usage of a reference architecture (we call this "success of a reference architecture") is determined by its design qualities and the "good" estimation of the assignees (i.e., the stakeholders

deciding to design the architecture) for the initiation of the design project in a proper context and with goals properly reflecting this context. Thus, for designing an effective architecture the assignees should consider not only the systems-to-be-designed when defining goals but also the context in which the architecture is defined and the goals of the architecture itself.

In our work, we have studied 16 reference architectures and observed that they have had different levels of success. We observed that due to the higher complexity of the design and application contexts in the case of reference architectures, certain architectures failed in defining business goals and elaborating architectures that "fit" their context. In this work, we investigate the contextual factors that influence the success of a reference architecture and the ways they influence it (see Figure 1, arrows indicate the causal effects between contexts, goals, and design). The results of this work provide a basis for the design of more successful reference architectures.

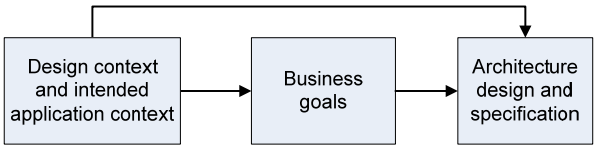

Figure 1. Relationship between context, goals, and design

In this paper, we first present a framework that allows reference architectures to be classified in three dimensions, according to their context, goals, and design. We call a reference architecture "congruent" if its goals are relevant for the context of the reference architecture and its design reflects the goals and context (illustrated by the arrows in Figure 1). We investigate which values in the dimensions can be combined for the design of a congruent reference architecture. Each tuple of values in our dimensions that defines a congruent architecture is called a "type". Reference architectures that fit into one of these main types have higher chances for success and effective application. Our framework together with the types of reference architectures can therefore serve as a "tool" for analysis of the chances for success of reference architectures. We illustrate the usage of this tool with two well-known reference architectures, i.e., the WRM [18] and the ANSI-SPARC DBMS architecture [28], and a number of other reference architectures.

The paper is structured as follows. In Section 2, we describe the framework for reference architectures. In Section 3, we present the main types of reference architectures. In Section 4, we discuss a number of reference architectures and relate them to our framework. Section 5 contains a discussion on related work. The paper ends with conclusions.

2. A framework for reference architectures

In this section, we present our framework for reference architectures. The framework is constructed with the aim to support analysis of reference architectures in terms of relationships between their context, goals, and architecture design/specification. Consequently, the framework is based on three dimensions: *context*, *goals*, and *design*. For each dimension, we define orthogonal sub-dimensions that address a specific aspect of the dimension. We have collected a set of possible sub-dimensions and have eliminated those that are non-orthogonal and those that have no correlation with any of the other dimensions. We have used the dimensions for concrete architectures defined in [15] and [16] for the definition of our sub-dimensions, selecting those dimensions that are relevant for our goals (e.g., we have omitted the "quality attribute" dimension which is out of the scope of our framework). In addition, we have used industry experiences and research results on classifications of reference architectures [4], [14], [21] for the identification of the sub-dimensions.

We denote the set of sub-dimensions by means of interrogatives in order to make them more intuitive for the reader (the usage of interrogatives is a well-known technique for high-level problem analysis [19]). We use the Where, Who and When interrogatives to address the context sub-dimensions, as these questions refer to the description of a contextual information. We use the Why interrogative to address the goal sub-dimensions. The How and What refer to operational aspects and are used for the design dimension.

Next, we discuss the sub-dimensions. Note that the values of a reference architecture can be mutually exclusive for some sub-dimension (i.e., a reference architecture can be attributed only one value from a sub-dimension) and mutually inclusive for other sub-dimensions (multiple values from the sub-dimension can be attributed to the architecture).

2.1. Context sub-dimensions (C)

In this dimension, we investigate aspects of the design and application contexts which may affect the business goals and design/specification of a reference architecture.

C1: *Where will it be used?*

In [15], a "stakeholder" dimension is defined that describes the stakeholders in the application context. We use this dimension to address on a coarse-grained level the organizations that are the intended recipients of the reference architecture. A reference architecture can be designed with an intended scope of a *single organization* or *multiple organizations* that share a certain property (they may share a market domain or a geographical property such as a country). Hence, the C1 sub-dimension contains two values: *single organization* and *multiple organizations*.

C2: *Who defines it?*

We use the "stakeholder" dimension from [15] as an inspiration to define its equivalent dimension in the "design process" context. The C2 sub-dimension lists the set of stakeholders that can be involved in the design of a reference architecture. At a high level, the stakeholders can be seen as the types of organizations involved in the process. These can be *software organizations, user organizations, research centres,* and *standardization organizations* (either commercial or government). These stakeholders may participate in the design process with people representing one or more of the following roles: *software designers, software users, software researchers, software/project managers*. As discussed in [8], each of these general roles can be further specialized into more specific roles (e.g., maintainers, administrators). However, for the goals of this paper such level of detail is not necessary.

C3: *When is it defined?*

The C3 sub-dimension addresses timing issues that may have an effect on the goals and design of a reference architecture. A reference architecture can be designed before any existing commercial system or a set of commercial systems together fully implement the reference architecture or after experience from commercial application has already been accumulated [4]. Hence, we define two possible values for this sub-dimension, i.e. *preliminary* and *classical* reference architectures. This dimension is inspired by the "transformation" dimension in [15].

2.2. Goal sub-dimensions (G)

Next, we investigate the goal sub-dimensions that may be affected by the context. One relevant sub-dimension is identified, i.e., intended usage of a reference architecture after its definition. Another possible sub-dimension is the "commercial" sub-dimension [14] which addresses the goal of the architecture in terms of commercial benefits (a reference architecture can be designed with the goal of delivering financial benefits for the assignee or with more "altruistic" purpose). However, in our work, we did not observe any conclusive relations of this sub-dimension with the context and design dimensions. That is why we omit it in our discussion.

G1: *Why is it defined?*

Based on [21], we distinguish two possible values for the intended usage of a reference architecture: *standardization* of concrete architectures (aiming at system/component interoperability) and *facilitation* of the design of concrete architectures (aiming at providing guidelines for the design of systems in the form of blueprints, patterns, etc.). Different types of standardization and facilitation goals can be distinguished [21]. However, these lower level goals do not interplay differently with the context dimensions (hence, they do not have to be addressed in our framework). This dimension is related to the "nature" dimension in [15].

2.3. Design sub-dimensions (D)

The sub-dimensions in this dimension describe a reference architecture in terms of its "operational" side, i.e., its design and specification. We have identified four relevant sub-dimensions that address the contents of a reference architecture, its level of detail, its level of concreteness, and the techniques used for its representation. The first sub-dimension corresponds to the *What* interrogative, the latter three together correspond to the *How* interrogative.

D1: *What is described?*

This sub-dimension lists the elements that can be defined in a reference architecture. It is based on the "type of information" dimension in [15], tailored for the context of reference architectures. As element types in a reference architecture, we distinguish *components*, *interfaces, protocols, algorithms,* and *policies and guidelines*.

D2: *How detailed is it described?*

This sub-dimension lists possible levels of detail at which the elements of a reference architectures (see D1) can be defined, i.e., it corresponds with an aggregation dimension. The sub-dimension is based on the "detail" dimension in [15]. Similar to [15], we distinguish three levels of detail, i.e., *detailed*, *semi-detailed*, and *aggregated* representation of the elements of a reference architectures.

D3: *How concrete is it described?*

This sub-dimension lists the possible levels of abstraction of a reference architecture. Abstraction is related to the level of choices made in an architecture in terms of technology, applications, vendors, etc. This sub-dimension is related to the "abstraction" dimension in [16]. We limit our values in this sub-dimension to *concrete, semi-concrete,* and *abstract* [16].

D4: *How is it represented?*

This sub-dimension lists the possible levels of formalization of reference architectures. It is based on the "representation" dimension in [15] and uses the same three levels defined there: *informal*, *semi-formal, formal.*

3. Types of reference architecture

The problem of identifying the types of reference architectures is a simple constraint-satisfaction problem in which combinations of values of the sub-dimensions should satisfy a number of constraints. The constraints that we have used for the selection of values of the sub-dimensions are postulates based on existing literature on reference architectures, our experience (including the one gained from the 16 reference architectures that we examined in our explorative study), and "common-sense" reasoning. These postulates are presented in this section as part of the description of the architecture types and the choices made in their construction. We have used a backtracking algorithm for the problem analysis. We started by fixing the value for the goal dimension and discussed the possible context and design values based on the goal dimension value. Next, we present our results on the types of reference architectures, first for the *standardization* goal value, then for the *facilitation* goal value.

3.1. Types of "standardization RA"

In this section, we take the case where the goal of a reference architecture is *standardization*. As already noted in the early ages of software architecting, an attempt to standardize architectures at an early stage is doomed to fail [28]. Thus, standardization reference architectures are typically *classical* reference architectures. Next, we sequentially "fix" the possible values for the C1 sub-dimension ("Where") and discuss the effect of this selection on the possible values for the other sub-dimensions. We start with fixing the values in the C1 sub-dimension because it has only two, mutually-exclusive values.

Table 1. Type 1

Dimension	***Values***
G1: Why	Standardization
↓	↓
C1: Where	Multiple organizations
C2: Who	User, software, and standardization organizations
C3: When	Classical
↓	↓
D1: What	Components, interfaces
D2: How	Aggregated components; (semi) detailed interfaces
D3: How	Abstract
D4: How	Semi-formal

Type 1: Reference architectures from Type 1 are *classical*, *standardization* architectures designed to be implemented in *multiple organizations* (see Table 1). Representative sets of *user* and *software* organizations should be involved in the architecture definition as standardization requires a consensus in order to be successful. The presence of a *standardization organization* facilitates the attainment of consensus and the establishment of the architecture as a standard. Due to the classical nature of the architecture, no research organizations are required (individual researchers may be involved by providing survey and summarization data from the field). Reference architectures from Type 1 contain a description of the *components* and *interfaces* as these are the elements that are target of standardization [28]. Note that specification of other elements than these is not excluded, but will not contribute to achieving the main goal of an architecture and may even have a negative effect on it (by providing unnecessary information and decreasing the efficient adoption of the architecture). Components are defined at a *high level of aggregation* as standardization of internal component details is unnecessary. Interfaces are defined at more detailed levels as their precise specification is crucial for achieving interoperability. Type 1 reference architectures are *abstract* (i.e., its elements are defined at a high level of abstraction) as each organization has the freedom to select the concrete implementation details according to its own settings. Type 1 architectures are *semi-formal* in order to provide a clear standard specification and to allow stakeholders who typically are inexperienced in strong formalization techniques to understand them. Example reference architectures from Type 1 are the WRM [18]

(discussed in detail in Section 4.1), the OSI RM [31], OATH [23], and CORBA [24].

Table 2. Type 2

Dimension	***Values***
G1: Why	Standardization
↓	↓
C1: Where	Single organization
C2: Who	Software users, designers and managers from the organization
C3: When	Classical
↓	↓
D1: What	Components, interfaces, policies / guidelines
D2: How	Aggregated, semi-detailed, detailed
D3: How	Semi-concrete, concrete
D4: How	Semi-formal

Type 2: Reference architectures from Type 2 are *classical*, *standardization* architectures designed to be implemented in a *single organization* (see Table 2). These reference architectures are designed to serve as a standardization tool for the design of a set of software solutions within the organization. For the same postulates as in the case of Type 1 architectures, representatives of all stakeholders should be involved (in this case, the potential *software users* and *designers* from the organization). Note that the software designers may be working on a temporary basis for the organization, e.g., on a consultancy project basis. *Managers* perform the role of the standardization organization in Type 1. Analogously to Type 1 architectures, reference architectures from Type 2 define architectural *components* and *interfaces* and are *semi-formal*. In addition, concrete organization-specific *policies* and *guidelines* are defined that will facilitate the usage of the architecture in the organization. The elements of reference architectures from Type 2 can be defined at *any level of aggregation* depending on the specific organization context. These architectures make certain choices in terms of technology, applications, and standards as a consequence of their standardization goals within the concrete organization. That is why they are *concrete* or *semi-concrete* reference architectures. An example Type 2 reference architecture is the Fortis Bank Reference Software Architecture[1].

A special case of Type 2 reference architectures can be observed in situations where they are used within software production organizations. In this case, reference architectures are designed to serve as the core of software product line architectures which are used in the software production process [25]. Although user organizations are outside the organization borders, they should still be involved in the design process. We see this case as a variation of Type 2 reference architectures (variation occurs in C2). Examples for this variation of Type 2 reference architectures are provided in [7].

[1] Due to their proprietary character, several of the architectures discussed in this paper are not published.

3.2. Types of "facilitation RA"

In this section, we discuss the case where the goal of a reference architecture is *facilitation*. We sequentially "fix" the two possible values for the C3 and C1 sub-dimensions and discuss the effect of this selection on the possible values for the rest of the sub-dimensions. The C3 and C1 sub-dimensions are chosen as a next step for a pragmatic reason, i.e., each of them has two, mutually-exclusive values. We start with the "*classical*" value of the C3 sub-dimension and the "*multiple organizations*" value of the C1 sub-dimension.

A classical, *facilitation* architecture designed for *multiple organizations* by multiple software and user organizations naturally becomes a standardization effort (see Type 1). A design effort conducted by one or more *user organizations* would face lack of capabilities, resources, and motivation. A design effort conducted by one or more *research organizations* may have the capability and resources for defining a *classical*, *facilitation* reference architecture for a domain but will lack the background for correctly addressing all practical requirements. Dissemination of the results is difficult in this scenario. Furthermore, such a project will not pose substantial research challenges which may result in the lack of motivation in the research organizations and long-term support for the architecture. The inclusion of preliminary elements in the architecture will be natural for a research organization and will be in violation with the classical character of the architecture. These issues remain even in the case of involvement of user organizations in the effort. Thus, although possible, the combination of research organizations(s) with multiple user organizations will face substantial hindrances. That is why we do not discuss it as a separate type. A *software organization* has the capabilities and resources to design a *classical*, *facilitating* reference architecture for a domain. Furthermore, it has a direct contact with user organizations and has the possibility to obtain requirements from them and to distribute the architecture among them. However, its incentive to

design a reference architecture will only be for promoting its own products. This leads us to the definition of the Type 3 reference architecture.

Table 3. Type 3

Dimension	***Values***
G1: Why	Facilitation
↓	↓
C1: Where	Multiple organizations
C2: Who	Software and user organizations
C3: When	Classical
↓	↓
D1: What	Components, interfaces, guidelines
D2: How	Aggregated components, interfaces; (semi) detailed guidelines
D3: How	Concrete
D4: How	Semi-formal

Type 3: Reference architectures from Type 3 are *classical*, *facilitation* reference architectures for *multiple organizations* designed by a *software organization* in cooperation with *user organizations* (see Table 3). Type 3 reference architectures are designed to promote a software product of the designing organization by describing its main *components* and *interfaces* and providing *guidelines* for their implementation. As a result, these reference architectures are *concrete* (promoting the organization technology), with *aggregate* components and interfaces (as they are aimed at a large set of contexts and complex details are hidden from clients) and detailed guidelines (needed to facilitate their implementation). Similar to Type 1 architectures, Type 3 architectures are *semi-formal* as they should be easily understood but still provide a clear specification of the architecture. Example reference architectures from Type 3 are Microsoft Application Architecture for .Net [20], and IBM PanDOORA.

Type 4: Reference architectures from Type 4 are *classical*, *facilitation* architectures designed to be implemented in a *single organization* (see Table 4). They are similar to Type 2 architectures but are designed only as a facilitation (guidance) tool in the design and implementation of systems in the organization. Due to their similarity to Type 2 architectures they need a similar stakeholder representation. Their facilitation role makes their preferred representation to be *semi-formal* or even *informal*. An *aggregated* or *semi-detailed* component design suffices for achieving the architecture facilitation goal. As they are designed for a concrete organization, they can be technology independent (*abstract*) or indicate technology choices (*semi-concrete*, *concrete*) Examples of software reference architectures of Type 4 are the Achmea Software Reference Architecture [13] and the ABN-AMRO Web Application Architecture [12].

Table 4. Type 4

Dimension	***Values***
G1: Why	Facilitation
↓	↓
C1: Where	Single organization
C2: Who	Users, designers, and managers from the organization
C3: When	Classical
↓	↓
D1: What	Components, policies / guidelines
D2: How	Aggregated/semi-detailed comp.; (semi) detailed guidelines
D3: How	Abstract, semi-concrete, concrete
D4: How	Semi-formal, informal

Next, we investigate the "*preliminary*" value of the C3 sub-dimension for *multiple organizations*.

Table 5. Type 5

Dimension	***Values***
G1: Why	Facilitation
↓	↓
C1: Where	Multiple organizations
C2: Who	Research centres, software design and user organizations
C3: When	Preliminary
↓	↓
D1: What	Components, algorithms, protocols
D2: How	Detailed, semi-detailed
D3: How	Abstract
D4: How	Formal, semi-formal

Type 5: Reference architectures from Type 5 are *preliminary*, *facilitation* architectures designed to be implemented in *multiple organizations* (see Table 5). They are designed to facilitate the design of architectures of systems that will become needed in the future. As these architectures are preliminary, *research centres* are typically leading the design effort. In order to address the *user* and *software* design requirements, *organizations* representing these roles should be involved in the design process as well. These reference architectures are innovative in their nature and have to define the *components* required in a system implementing it, *algorithms* that can be used to support the operation of the components, and *protocols* that demonstrate the interactions among the components.

These elements have to be *detailed* as they have to provide details for the innovative aspects they define (showing implementability of components, clarifying their operation, etc.). Reference architectures from Type 5 abstract from a concrete technology, as it may still not exist or be immature (hence, they are *abstract* architectures). Unambiguousness and evidences for their qualities are important to convince the usage of the architecture for the design of the first systems from this class. That is why they are *formal* or *semi-formal* architectures. The ANSI-SPARC database reference architecture [28] is an example of a reference architecture that closely resembles a Type 5 architecture.

We view *preliminary*, *facilitating* reference architectures designed for *multiple organizations* by only a *research centre* as a special case of Type 5 architectures. The origin in pure research environment of these reference architectures results in "futuristic" designs that do not concentrate on the requirements of the domain stakeholders but on the innovative elements of the architecture. That is why these architectures are usually not considered for system implementations in practice. Their main contribution is in inspiring future research efforts in the domain (depicting main software issues, providing blueprints for prototype implementations, etc). The success of these architectures is hard to estimate. Examples for this specialization of Type 5 architectures are ERA [2], AHA [30], and eSRA [22].

Design of a *preliminary*, *facilitating* reference architectures for a *single organization* is plausible. However, the effort of defining a preliminary reference architecture for a single organization requires substantial resources. Only leading organizations might invest in visionary architectures. As we did not find an existing example for such reference architectures, we do not define it as a separate type.

4. Analysis of reference architectures

In Section 3, we have discussed the types of reference architectures in which goals, context, and design are in congruence and provided example references for each of them. In this section, we present the WRM [18] as an example of a reference architecture that fits in a type and discuss it in detail. The ANSI-SPARC database reference architecture [28] is presented as an example architecture that does not completely fit in one of our types. We discuss the consequences from this misalignment for its level of success. We have selected these two reference architectures because of their popularity and the possibility to evaluate their contribution to the design of concrete architectures from the perspective of time. In addition, we briefly analyse the position of a number of less known reference architectures in our framework and their level of success. The findings for all reference architectures that we studied are summarized in a table.

4.1. The Workflow Reference Model

The Workflow Reference Model [18] is an example of a Type 1 reference architecture. It was designed by the Workflow Management Coalition - a standardization consortium of user and software organizations [3]. Their goal was to elaborate a standard for the design of workflow management systems. By the time of the definition of the WRM, substantial experience with workflow management systems had been accumulated. The WRM defines the system components and the interfaces between them on a high level of aggregation. There are no references towards a specific technology in the WRM.

The congruence between the goals of the designers, context, and architecture design made WRM a successful reference architecture. It is a well-known architecture that has been used as a basis for the design of numerous concrete architectures of workflow management systems [10].

4.2. The ANSI-SPARC DBMS reference architecture

The ANSI-SPARC DBMS reference architecture was conceived as a standardization architecture for multiple organizations [28]. The design team was composed of user, software, and research organizations. The architecture specifies components and interfaces. It is abstract, semi-detailed, and semi-formal. Based on these values, the architecture can be classified as Type 1 architecture with a variation in the C2 sub-dimension (involvement of researchers). However, as its designers concluded 'post-factum', the existing technology was not able to support it [28]. In the newly defined context, the architecture became preliminary, facilitation architecture positioning itself as a Type 5 reference architecture. The mismatch of its values and Type 5 values occurs in the D1 sub-dimension, which as a result of the initial confusion of goals and context of the architecture, defines components and interfaces instead of components and algorithms. This misalignment decreased the effectiveness of the architecture as the support for certain components had to be further investigated beyond this design effort (which has significantly

delayed the usage of the complete architecture in practice). However, the congruence of goals, context and the other design dimensions (as defined in Type 5) contributed to the success of the design principles of the ANSI-SPARC architecture (well-known as the "ANSI-SPARC three-layer model") which became a fundamental model for the design of database management systems.

4.3. Other reference architectures

In this section, we discuss a number of less known reference architectures. We provide a summary of our findings for the 16 reference architectures studied by us in Table 6.

The Workflow Management Systems Reference Architecture (WMS RA) [17] is a classical, facilitation architecture designed for multiple organizations. It is abstract, detailed, semi-formal, describing system components and their operation. Based on its goals and context it resembles a Type 3 reference architecture. However, it was mainly designed in a research centre, and thus, differs from a Type 3 architecture in its C2 sub-dimension (and consequently in the D2 and D3 sub-dimensions). As discussed in Section 3.2, the origin of the architecture in a research centre led to the lack of attention by the domain to the WMS RA and it was never used as a basis for the design of a concrete architecture. The success of the WMS RA could have been greater if it was conceived as a Type 1 architecture by involving additional stakeholders and becoming a standardization effort (and a competitor of the WRM).

The INAHL reference architecture [29] is a recently published, classical, facilitation reference architecture designed for multiple organizations in the petroleum industry. It was designed by a research centre in cooperation with one user organization [5]. The architecture is detailed, concrete, semi-formal, describing mainly the system components. Thus, it resembles a Type 3 architecture but similar to the WMS RA, it was conceived in a research environment. In contrast to the WMS RA, it involved a user organization and is a concrete architecture. This makes it a better fit for the Type 3 than the WMS RA. In our opinion, the lack of a constant support for the INAHL RA by a software organization (as required in Type 3) will be a reason for its limited success. So far, it has been experimentally implemented in the user organization but is no longer in practical use there [5].

The Achmea Software Reference Architecture [13] was developed to guide the application developers at Achmea in the development and integration of new applications. It is designed by a software organization (IBM), together with designers from the user organization (Achmea). It is a classical reference architecture as it makes use of IBM best-practices, supplemented with best-practices from Achmea. It contains a description of components and a number of organization-specific policies. The architecture is concrete as it addresses specific software products and protocols. The architecture is aggregated, providing only high-level clustering guidance. The architecture is described in structured natural language (informal). We conclude that this is a Type 4 architecture with congruent goals, context, and design. The architecture has been used in numerous projects within Achmea.

Table 6. List of studied reference architectures

	T	G1	C1	C2	C3	D1	D2	D3	D4	LS
WRM [18]	1	X	X	X	X	X	X	X	X	4
OATH [23]	1	X	X	X	x	x	X	X	X	*3*
OSI RM [31]	1	X	X	X	X	X	X	X	X	4
HIF [27]	1	X	X	X	X	x	X	X	X	3
CORBA [24]	1	X	X	X	X	X	X	X	X	4
FORTIS RSA	2	X	X	X	X	X	X	X	X	4
IBM PanDOORA	3	X	X	X	X	X	X	x	X	3
WMS RA [17]	3	X	X	-	X	x	-	-	X	1
INAHL [29]	3	X	X	x	X	x	X	X	X	*2*
MS .NET [20]	3	X	X	X	X	X	X	X	X	4
ACHMEA RA [13]	4	X	X	X	X	X	X	X	X	4
ABN WAA [12]	4	X	X	x	x	X	X	X	X	2
ANSI-SPARC [28]	5	X	X	X	X	x	X	X	X	3
ERA [2]	5	X	X	x	X	x	X	X	X	-
eSRA [22]	5	X	X	x	X	x	X	X	X	-
AHA [30]	5	X	X	x	X	X	x	X	X	-

In Table 6, we list the values for all reference architectures that we studied. A row in the table shows the type that the reference architecture resembles the most (indicated in the "T" column) and its values for each of the sub-dimensions in our framework. We use "X" to denote a complete match between the values of the architecture and those recommended in the type, "x" that certain variations occur, and "-" that there is no match between them. The last column ("LS") indicates on a scale from 1 to 4 (1 being the lowest) our estimation for the level of success of the architecture. As already discussed, we do not estimate the architectures from the variation of Type 5.

Our measurement of the success of a reference architecture is based on the acceptance of the architecture by the domain community. Acceptance by the community leads to a higher number of applications of the reference architecture in the design of concrete architectures. Our estimations for the success of the reference architectures presented in Table 6 that exist relatively long are based on existing publications on their usage [10] (for WRM), [9] (for

HIF) and on our experiences and judgement for the acceptance of the architectures by the domain communities (for the OSI RM, CORBA, ANSI-SPARC DB, FORTIS RSA, IBM PanDOORA, WMS RA, MS .NET, ACHMEA RA, ABN WAA). For architectures that were released recently, i.e., OATH, INAHL, ERA, eSRA (shown in italic in Table 6), no conclusive evidence for their usage can be provided. Our estimation for them is based on their match with the indicated type and can be seen as a prediction for their level of success. Clearly, our estimations are not based on precise measurement criteria. Characteristics of an architecture can affect its explicit usage. For example, copyright issues may be a reason for not stating the usage of a reference architecture. Providing stronger evidence for the level of acceptance of the reference architectures and details for the correlation between the congruence of *goals*, *contexts*, and *design* of the reference architectures and their acceptance is future work. We use our estimations only as basic indications for the existence of this correlation.

5. Related work

The literature on reference architectures is scarce. Definitions and brief explanations on reference architectures are provided in [6] and [26]. In [14], the authors acknowledge the lack of clear understanding of reference architectures and provide a multi-dimensional classification of reference architectures. The classification is, however, ad-hoc and mainly based on practical experiences. In [21], an overview of reference architectures is presented. Similar to [14], the authors concentrate on the description of the many facets of reference architectures. The goal of the paper is to "create guidelines for the content of a reference architecture and the process to create and maintain it". The dimensions discussed in [21] are less explicit compared to our work and have a descriptive goal. The dimensions discussed in our paper are clearly structured and are defined to serve as a framework for the classification of the level of congruence of goals, context, and design of reference architectures. In [21], the authors acknowledge the need for congruence of different dimensions for the definition of successful reference architectures but do not state requirements with respect to these dimensions.

The design of reference architectures with congruent business goals, functionalities and qualities for software product line architectures is addressed in [25] and [1]. However, this work is limited for the concrete context of product families discussed in Type 2 reference architectures (see Section 3.1).

As already discussed, the quality of a reference architecture is clearly also a factor for its success. Evaluation of reference architectures for their qualities is performed through the Architecture Tradeoff Analysis Method (designed for evaluation of software architectures) [6], [8], [11]. The limitations of ATAM for the context of reference architectures are addressed in [4], where extension and tailoring of ATAM for the context of reference architectures are proposed.

6. Conclusions

In this paper, we present a three-dimensional framework for the classification of reference architectures. The three dimensions are based on the need for congruent goals, context and design of reference architectures. We use the framework to define five types of reference architectures that have congruent values in the three dimensions. The match of a reference architecture with one of these types is a pre-condition for its effective usage in the design of concrete architectures. The five types are defined by using existing publications, reasoning on the possible combinations of values, and by investigating 16 reference architectures and estimating their success. We illustrate our conclusions by positioning the reference architectures studied by us in our framework and discussing their level of success.

The framework that we present in this paper can be used as a tool for the analysis of existing reference architectures and identification of the reasons for their level of success. More importantly, the framework can be used as a tool before and during the definition of reference architectures, indicating potential deviations in their congruence between goals, context, and design. We believe that this work will contribute to the design of more successful reference architectures and in general to the structuring of the conceptual space of reference architectures.

The main limitation of this study is the relatively small set of reference architectures that was used in our exploratory study and in the design of the architecture types. Validating and extending our results with more reference architectures will improve and strengthen our conclusions. We see this as a next step in our research.

References

[1] M. Anastasopoulos, J. Bayer, O. Flege, and C. Gacek, "A Process for Product Line Architecture Creation and Evaluation: PuLSE-DSSA," IESE-Report 038.00/E, 2000.

[2] S. Angelov and P. Grefen, "An E-contracting Reference Architecture," The Journal of Systems and Software, vol. 81, no. 11, Elsevier, 2008, pp. 1816-1844.

[3] S. Angelov and N. Palmer, Pers. communication, 2008.

[4] S. Angelov, J. Trienekens, and P. Grefen, "Towards a Method for the Evaluation of Reference Architectures: Experiences from a Case," in *Software Architecture, 2nd European Conf., ECSA 2008,* Springer, 2008, pp. 225-240.

[5] S. Angelov and G. Urdaneta, Pers. communication, 2009.

[6] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, Addison-Wesley Prof., 2003.

[7] J. Bayer, D. Ganesan, J.-F. Girard, J. Knodel, R. Kolb, and K. Schmid, "Definition of Reference Architectures based on Existing Systems," Fraunhofer IESE, IESE-WP2.6, 2003.

[8] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*, Addison-Wesley Professional, 2002.

[9] F. Ferrara, "The standard 'Healthcare Information Systems Architecture' and the DHE middleware," *Int. J. of Medical Informatics*, vol. 52, no. 1, pp. 39-51, 1998.

[10] L. Fischer, *Workflow Handbook 2004*, Future Strategies Inc., 2004.

[11] B. Gallagher, "Using the Architecture Tradeoff Analysis Method to Evaluate a Reference Architecture: A Case Study," SEI, Carnegie Mellon University, CMU/SEI-2000-TN-007, 2000.

[12] D. Greefhorst, "Een applicatie-architectuur voor het web bij de bank — de pro's en contra's van toestandsloosheid," Array Publications, 1999 (in Dutch).

[13] D. Greefhorst and P. Gehner, "Achmea streamlines application development and integration," Via Nova Architectura, 2006.

[14] D. Greefhorst, P. Grefen, E. Saaman, P. Bergman, and W. van Beek, "Referentie-Architectuur: Off-the-Shelf Architectuur," in *Landelijk Architectuur Congres 2008*, NAF & SDU, Nieuwegein, The Netherlands, 2008 (in Dutch).

[15] D. Greefhorst, H. Koning, and H. van Vliet, "The many faces of architectural descriptions," Information Systems Frontiers, vol. 8, no. 2, 2006, pp. 103-113.

[16] P. Grefen, *ICT Architectures*, Eindhoven University of Technology, Eindhoven, 2008.

[17] P. Grefen and R. Remmerts de Vries, "A reference architecture for workflow management systems," Data & Knowledge Engineering, vol. 27, no. 1, 1998, pp. 31-57.

[18] D. Hollingsworth, "The Workflow Reference Model," Workflow Management Coalition, TC00-1003, 1995.

[19] M. Metcalfe, *Reading Critically at University*, Sage Publications Ltd, 2006.

[20] Microsoft, "Application Architecture for .NET: Designing Applications and Services," Microsoft, 2002.

[21] G. Muller, "A Reference Architecture Primer," Eindhoven Univ. of Techn., Eindhoven, White paper, 2008.

[22] A. Norta, *Exploring Dynamic Inter-Organizational Business Process Collaboration,* PhD Thesis, Eindhoven University of Technology, Eindhoven, 2007.

[23] OATH, "OATH Reference Architecture, Release 2.0," Initiative for Open AuTHentication (OATH), 2007.

[24] OMG, "Common Object Request Broker Architecture: Core Specification," OMG, Inc., Version 3.0.3, 2004.

[25] M. Pinzger, H. Gall, J.-F. Girard, J. Knodel, C. Riva, W. Pasman, C. Broerse, and J. G. Wijnstra, "Architecture recovery for product families," in *Software Product-Family Engineering,*Berlin: Springer, 2004, pp. 332-351.

[26] P. Reed, "Reference Architecture: The best of best practices," 2002.

[27] N. Saranummi, M. Demeester, A. F. P. d. Talens, J. Harrington, V. Heimly, J. M. d. l. Riva Grandal, and J. Taylor, "Healthcare information framework," Int. Journal of Bio-Medical Computing, vol. 39, no. 1, 1995, pp. 99-104.

[28] SPARC-DBMS Study Group, "Interim Report: ANSI/X3/SPARC Study Group on Data Base Management Systems 75-02-08," FDT, vol. 7, no. 2, 1975, pp. 1-140.

[29] G. Urdaneta, J. Colmenares, N. Queipo, N. Arapé, C. Arévalo, M. Ruz, H. Corzo, and A. Romero, "A reference software architecture for the development of industrial automation high-level applications in the petroleum industry," Comp. in Industry, vol. 58, no. 1, 2007, pp. 35-45.

[30] H. Wu, *A Reference Architecture for Adaptive Hypermedia Applications,* PhD Thesis, Eindhoven University of Technology, Eindhoven, 2002.

[31] H. Zimmermann, "OSI reference model - the IS0 model of architecture for open systems interconnection," IEEE Transactions on Communications, vol. 28, no. 4, 1980, pp. 425-432