The Integrated Architecture Framework Explained

Why, What, How

Bearbeitet von
Jack van't Wout, Maarten Waage, Herman Hartman, Max Stahlecker, Aaldert Hofman

1st Edition. 2010. Buch. XVIII, 246 S. Hardcover
ISBN 978 3 642 11517 2
Format (B x L): 15,5 x 23,5 cm
Gewicht: 558 g

Chapter 2
IAF's Architecture

2.1 Introduction

Explaining IAF must start with the basics, the core elements of the framework. Actually, our aim is that you understand IAF's own architecture. Once you understand IAF's structure and underlying ideas it will be easier to apply it to the specific situation you are in. This chapter is all about helping you understand the architecture behind IAF.

IAF applies the basics of general construction methods. All construction methods have a common approach. They address questions in a specific order. The order is expressed as interrogative pronouns: Why, what, how, with what. There are reasons for the sequence of the questions.

If you don't understand why you need to do something, you can't work out what needs to be done. You have to understand the context before you can start working effectively. Studying the context helps define the scope and objectives, and thus helps with staying focused. So understanding 'why' is the first thing to do.

If you don't understand what needs to be done (the requirements), you will never be able to craft the solution. You have to define the requirements before creating the solution. Once you understand the requirements you are often able to produce a concept of what is needed. So after understanding why you are doing something you need to address the 'what' question by defining the requirements (both functional and non-functional) and getting a concept of the solution.

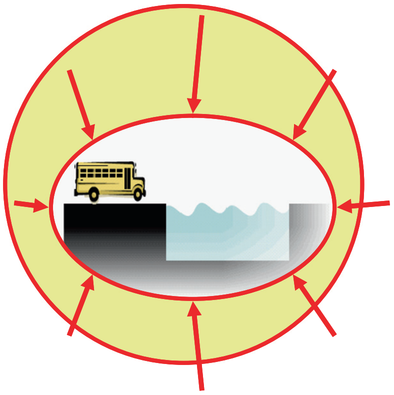
With a good understanding of the requirements you should be able to design a logical solution to the problem, answering the question 'How will the solution look like'. In other words you structure the solution. You create the solution on a logical level to enable flexibility. All architectures will take time to implement. Circumstances as well as real life physical solutions change over time. By describing the architecture on a logical level you will be able to deal better with new insights and adapt it at the moment you physically implement a specific part of the architecture.

J. van 't Wout et al., *The Integrated Architecture Framework Explained*, 5

Finally you can decide with what physical things the solution can be realized. This implies allocation of physical things to the logical solution.

A practical example will demonstrate how this all works.

Imagine you are living in a city that has been built on both banks of a large river. There are a number of bridges crossing the river, but traffic has grown through time and there are constant traffic jams. In two years time the city is planning to host a big cultural event, and it wants to show a modern, young city that is well prepared to handle all the tourists that are visiting the event. So, the scope and context of the problem become clear. A new bridge needs to be built in the city within 2 years and it should reflect the 'young, modern' look that the city wants. The 'why' question should be clear by now.



Now we need to address the 'what' question. We need to find out what the bridge has to do. Does it have to carry pedestrians? Does it have to carry cyclists? Does it have to carry cars and trucks? Does it have to carry trains? How many of each type does it have to be able to carry during average and peak hours of usage? Which types and numbers of boats have to be able to pass under the bridge? What is the river's required water flow capacity and what would happen if that were reduced if the bridge were to have many pillars? What does the patron want it to look like to reflect the young, modern look? Once these types of questions are answered we have a clear understanding of the 'what' question. We know the requirements and are able to craft the solution.

The third main step is creating a logical model of the solution, thus answering the 'how' question. Now there is a slight snag here. Real life has shown us that there never is one solution. There always choices, and all solutions are trade-off of different aspects like cost and performance. So, in effect, if you think there is only one solution, you might have to think a little longer. There are solution alternatives that have to be considered. In this case we can identify different types of bridges that could provide sufficient river-crossing capabilities. Some of them will fit the principles we have defined regarding 'young and modern' better than others. Others might not be able to be built within 2 years. We could even

consider a temporary bridge, which would probably make a financially interested stakeholder happy, could be finished in a short period of time, avoids difficult technological studies on water flows but the patron who dreams of grandeur and eternal fame would be somewhat disappointed. In the end we present the best alternatives to the stakeholders so they can choose.

Now real life physical architecture can start. We choose the materials we need for each part of the bridge. By this time the cooperation with various construction engineering disciplines has begun. First we jointly determine what is needed for the foundation. Then we select the rebar materials and concrete. We define which steel is needed for the deck. We create the specifications the construction teams need to create the design of the bridge. We create other visualizations of what the real life solution will look like. The 'with what' question has been answered. The architecture has been designed. It can be handed over to the engineers who are actually going to build the bridge. The architect will remain involved in order to solve issues that come up during the construction phase and which might affect the starting points or principles that underlie the entire idea behind the bridge.

We are going to explain IAF's own architecture using the 'why-what-how-with what' approach explained above.

2.2 The Context: The "Why" of IAF

2.2.1 Vision

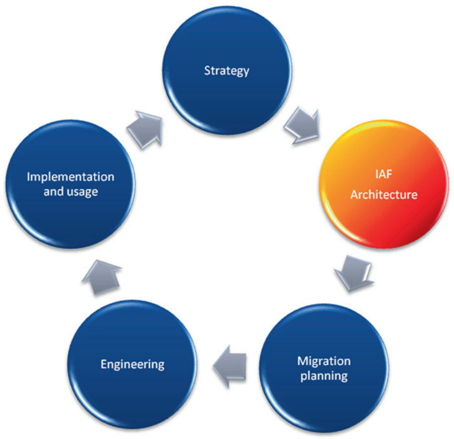
The vision we had when we started to develop IAF consisted of a number of elements. The most important goal was that we wanted to be able to provide world-class services to our clients, and were convinced that architecture was key in this. The ever increasing complexity and risk in the engagements we were working on made this obvious. We also needed a robust and mature toolset to deliver a constant quality of architecture services and a consistent experience where a client is engaged many times over a period of time. The toolset had to successfully address the alignment of business and IT. It had to be independent of a specific architect: the way we work and approach architecture should be common across all architects. In this perspective we now speak of IAF as a 'design school', with very specific style, approach and characteristics that we feel make a difference: for us as Capgemini being a global company delivering services, and for our clients ranging in size from the Fortune-500 to the local medium enterprise. In addition to this it had to provide a platform on which we could expand and improve the architecture profession. We decided up-front that it had to be based on real-life experience, and not on pure theory. We knew that we were embarking on a journey with an uncertain destination, and wanted to base that journey on things that had been proven in the field.

2.2.2 Scope

The original slogan we used in regard to IAF was: 'For a system to work as a whole, it must be architected as a whole'. This slogan nicely depicts many elements that are part of the IAF scope.

The first striking word is 'system'. What do we mean by 'system'? We have small projects that upgrade existing applications to very large programs in which we support complete post merger integrations of global companies or companies constantly reinventing themselves. All these types of projects potentially require some form of architecture. They all create or change a 'system'. Thus IAF's scope needs to cover all types of architecture engagements at all levels in an organization. Enterprise level – spanning business units –, Business unit (business domain) level and Solution level. Enterprise and Business unit level are commonly meant for supporting planning activities. Solution level is aimed at guiding the engineering of the solution.

The second important word in the slogan that is related to IAF's scope is 'whole'. Capgemini always approaches IT from the viewpoint that it has to support the business. This implies that architecture should always be justified in business terms and traceability to business requirements. Even when we are architecting a purely IT system, the business should provide the objectives and drivers for the architecture.



The third word in the slogan that is related with scope is 'architected'. IAF is aimed at the architects profession, and should only describe things for which the architect has the main responsibility. Therefore IAF does not provide support for the creation of an organization's vision, mission and strategy. These are defined as input for IAF. Some basic input that the architect usually derives from the vision, mission and strategy is contained in IAF to assist architects in collecting input if it is not there. On the other end, migration planning has also been put out of scope of IAF, because that activity is not the architect’s main responsibility. In general, migration planning is a joint exercise with the engagement manager as the responsible person. IAF also tries to avoid overlap with other professions like business analysis and engineering. There are many touch points between the architects and these professions, just like there are many touch points between architects and engineers in real life construction architecture. This creates a gray area. Where does the architect stop and the engineer start? The most pragmatic answer to this question is that there is a difference in focus. The architect focuses more on how the ‘system’ fits into its environment. The engineer focuses more on the internal structure of the ‘system’, within the boundaries that have been set by the architect. In other words: the architects focuses on the behavior and non functional requirements (‘black box’) while the engineer on the internal construction (‘white box’).

2.2.3 Objectives

Through time IAF’s objectives have not changed much. The framework has evolved due to increased understanding of architecture in the IT industry as a result of the pursuit of the objectives.

IAF’s core objective is to provide a common way of architecting. Originally it was intended to do this within Capgemini. Nowadays more and more organizations also adopt this common way of working.

IAF also must provide a communication framework to achieve the common way of architecting. This ‘common language’ needs to be adopted throughout our organization and across the regions where we operate, particularly when serving global clients such as General Motors. This objective has proven to be difficult to realize – but we feel we succeeded. One word can have the same definition in multiple countries and still be perceived to be different. An example that is popular in Capgemini is the confusion we had around the term ‘Business event’. In the Netherlands this was perceived as ‘something that *can* happen in an organization’. In the UK it was perceived as ‘something that *has* happened in an organization’. Therefore it was very understandable for the Dutch to propose basing a to-be architecture on business events. The UK architects tended to disagree. Their counterargument was: ‘How can you base a to-be architecture on something that has happened in the past?’

Sometimes we have even invented new words to resolve terminology discussions. A nice example is the term ‘archifact’ that was used in IAF version 3 because we could not come to an agreement at that time on the usage of the term 'artifact'. Another example is the term 'scenario' which we later replaced by the term 'solution alternative'. Many US architects were confused with the term 'scenario', as they associated this term with the movie industry.

The common way of working and common language were enablers for the third objective that was defined by Capgemini. We have always had the need to staff large projects from around the world. One of the challenges in doing this was getting the right person with the right skills in the right place. IAF has proven to be a great help in achieving this objective specifically due to the common language.

The fourth main objective also comes from the large projects we work on. Managing their complexity and thereby reducing project risk is important for Capgemini because it raises the quality level of the results we deliver to our clients. Clients have the same objectives.

2.2.4 Constraints

One of the major constraints for IAF is related to its scope. IAF focuses on things that are the architect's main responsibility. Topics, that (a) we assist in and (b) are the main responsibility of other roles in the project, are not to be positioned within IAF. They will be covered in frameworks that the other roles use to standardize and professionalize their work.

Another constraint we have implicitly used in the development of IAF is the focus on business and IT. Real life has proven that any change in an organization should be realized by addressing a large number of topics. An acronym that is sometimes used to describe these topics is COPAFITHJ. In the preparation of any organizational change we should consider the *C*ommercial, *O*rganizational, *P*ersonnel, *A*dministrative, *F*inancial, *I*nformation processing, *T*echnological, *H*ousing and *J*udicial impact of that change. In line with the mission of Capgemini, IAF focuses on process, information and technology. However the basic structure within IAF can be used to extend the topics that are addressed. For example one client wanted to add a 'product architecture' to the framework to address commercial aspects while another client wanted a 'financial architecture' to address financial aspects.

2.3 Requirements: The 'What' of IAF

The vision, scope, objectives and constraints mentioned in the prior sections are the basis for the requirements that have been defined for IAF. This section describes the requirements behind IAF. We have chosen an informal descriptive approach of defining them, in contradiction to IAF based architectures itself, in which requirements are documented in a formal and prescribed way. We have chosen the freedom to add examples or elaborate on a topic more than we do in an architecture to help you understand the background of the requirements. In each of the sections below we address a specific requirement of IAF.

2.3.1 Requirement: Understand, Structure and Document Architecture Input

An important aspect of IAF is the requirement to provide support to help the architect understand, structure, and document the input that is needed to create an architecture. This understanding, structuring, and documenting of input is important. Things like strategy, vision, context, and scope are never standardized products. Almost all consulting firms in that area have different approaches to the topics. So what IAF needs to do is to provide the architect with a checklist of things that can be used as input. Very often the input that is relevant for the architect is scattered in multiple (large) documents and needs to be derived. It is very inefficient to have all architects working on the architecture to read all the input documents. Collecting the relevant input and documenting it will speed up the 'on boarding' of architects. Topics that are to be part of the input are:

- The strategy and vision that the architecture has to support;

> *Facts: No strategy, no architecture. No vision, no architecture.*

- The scope of the architecture;

> *Fact: When you define and agree the scope with your principal, you demonstrate to the client that you really understand the problem. This frequently leads to a modified scope (larger, smaller, shifted) because you show what the actual, real problem is.*

- The context of the solution;

> *Fact*: When you clearly recap the context, more information about the scope will emerge.

- The objective of the architecture;

> *Fact: When you don't have a clear understanding of the objectives of the architecture (the question which the architecture will answer), you will not know when your architecture is good enough. A clear objective will prevent you from going into details that are not relevant from the perspective of the architecture.*

- The principles to be applied to the solution;

> *Fact: Without principles you will not be possible to design an architecture that satisfies all stakeholders. Principles also help you to identify and resolve conflicting requirements.*

- The current state which the architecture has to take into account.

> *Fact: You will almost never architect a green field solution, in which you will have no constraints from the current state on the architecture you are designing.*

2.3.2 Requirement: As Simple as Possible

Architecture in the IT industry needs to address complex problems. Commonly we encounter organizations with hundreds of business processes and thousands of applications. This means we have to tackle very complex problems. This must not lead to a framework that is more complex than needed to solve the problem.

2.3.3 Requirement: Split Complex Problems into Smaller, Resolvable Ones

Many architectures can become relatively large and complex. A tried and tested way to solve large and complex 'configuration problems' – which an architecture is – is to split the overall problem into smaller ones that can be resolved, as long as the identification of the smaller problems is in line with ideas that you have about the overall solution. IAF is required to support this approach as we typically design architectures in response to complex situations.

2.3.4 Requirement: Cover the Breadth and Depth of the Architecture Topics Needed to Support Capgemini's Mission and Vision

Architecture is a supporting function. It is not a goal in itself. IAF needs to support the Capgemini mission ('enabling transformation') and vision ('enabling freedom'). It is to enable its clients to transform and perform through technologies. Business and IT transformation services always have been Capgemini's main focus. Therefore IAF must support them.

2.3.5 Requirement: Support All Relevant Types of Architecture

Construction architecture recognizes types of architecture like city planning architecture, zone planning architecture and building architecture. We also recognize different types of architecture in the IT industry. The types most commonly identified are:

(1) Enterprise architecture, aimed at supporting enterprise wide decision making and planning, and shaping the enterprise landscape;
(2) Domain architecture, aimed at supporting business unit level decision making and planning, and shaping the domain landscape;
(3) Solution architecture aimed at providing architectural guidance to programs and projects;
(4) Software architecture, aimed at providing architectural guidance to software development.

As Capgemini provides services in all these areas we need to be able to deliver all types of architecture described above.

2.3.6 Requirement: Flexibility in Content

Not every type of architecture requires the same amount of detail when addressing a given topic. For example interfaces might only need to be identified at enterprise or domain level, while it is very relevant to specify them in detail when working on a solution or software architecture. Also the granularity of the architecture’s elements has to be variable. The topics we talk about at enterprise level (sales, marketing, HR production and finances) are often larger than the topics we talk about at solution level (order entry, stock level checking, order picking and order packing).

The content of IAF must be able to cope with these differences and should not depend on the industry sector where it is applied.

2.3.7 Requirement: Flexibility in Process

As Capgemini works for many different clients they encounter many different situations. Ask any Capgemini architect if they have used the exact same process to deliver an architecture twice. We promise you that they will say ‘no’ most of the time. For this reason we need to have process flexibility. In fact we actually need to split process and content. Often we need to be able to create similar content using different processes. We might be working together with Capgemini transformation consultants in one engagement and with consultants from another consulting firm in the other. Both groups have different approaches to transformation consulting that we have to cope with.

2.3.8 Requirement: Traceability and Rationalization of Architecture Decisions

Most architectures have to address a broad number of topics. Many topics influence each other. For example, centralizing the financial administration of 5 business units into 1 shared service center will influence the decisions on IT support that is required, which in turn will influence the infrastructure that is required. As architectures will have to be maintained over time it is very important to understand why a certain decision was made. Traceability of decisions, and the documentation of the rationales behind the decision is of paramount importance.

2.3.9 Requirement: Terminology Standardization

When you want to provide a common way of architecting, standardization of terminology used is mandatory, especially in regard to the terminology the architects use amongst each other. When they speak the same language, they can work together better. Through time Capgemini has learned that we commonly have to adapt our architecture terminology to terms that our clients have been using and are accustomed to. This in turn has led to the decision to introduce the usage of synonyms into IAF. Synonyms have formally been introduced into IAF version 4.5.

2.3.10 Requirement: Standardized Organization of Architecture Elements

Where possible we need to solve similar problems in similar ways, thus working toward the common way of architecting. By standardizing the organization of architecture elements, we will discover similar problems and enable ourselves to solve them in similar ways.

2.3.11 Requirement: Address Both Functional and Non-functional Aspects

Many architecture frameworks provide average to good support for structuring architecture from a functional point of view. They also commonly state that non-functional aspects must be taken into account. However they do not provide support for both functional and non-functional aspects. IAF must cover this, in order to develop balanced functional solutions that perform as desired; for instance, it makes a big difference if you need to support Straight Through Processing of 100 million orders a day as opposed to manually supporting 100 orders a day.

2.3.12 Requirement: Provide a Basis for Training New Architects

Development of a framework is just one step into achieving a common way of architecting. People will leave the company and new people will join. People will decide to develop their career toward architecture. IAF needs to provide a basis for its deployment by providing a basis for the creation of training material.

2.3.13 Requirement: Provide Sufficient Information for Engineers

If a building architect would deliver results that were insufficient for the engineers to build the building correctly, the architect would be sued. The same should go for IT architects. They have to deliver standards, rules, guidelines, and specifications in such a way that engineers can build the desired solution. An additional dimension for this requirement is that engineers are evolving their way of working, just as we are. When we started forming IAF, linear development was commonplace, and RUP was just starting to be used. Nowadays RUP is commonplace, and methods like XP and Agile are becoming more and more common. The different development methods have different input requirements. IAF has to cope with that.

2.3.14 Requirement: Provide Sufficient Information for Planners and Portfolio Managers

A city planning architect would also be fired if he could not deliver what the city planners needed. Enterprise and domain level architectures are mainly aimed at supporting planners and portfolio managers. Our output needs to address their input requirements. Here too we need to be flexible, as planning and portfolio management in the IT industry are not part of the business fabric yet.

2.3.15 *Requirement: Take Stakeholders and Social Complexity into Account*

Different stakeholders and their concerns need to be addressed and continuously managed as stakeholders have concerns with respect to the system architected. Depending on the stakeholder these concerns will differ. The architect will be working with a number of stakeholders, each with their own concerns on the result being architected. There is a need to not only consider one aspect area (such as business, applications, infrastructure, security, etc.) in isolation, but rather to see all aspect area as being part of an integrated whole. The concerns of stakeholders, especially when considered in parallel, are hardly ever limited to one aspect area only. Stakeholders will want to gain insight into these aspect areas, their interdependencies, and the possible impact of future developments on their concerns. We must be able to communicate with a possibly large and diverse group of stakeholders, addressing their individual concerns while at the same time working on a shared understanding and commitment of the result architected.

2.3.16 *Requirement: Enable a Sound Approach to Solution Alternatives*

Architecture in the IT industry is all about trade-offs. The best performing solution might not be secure enough, or it might cost too much. There is never one solution that perfectly answers all requirements. IAF has to provide an approach to define and compare different solution alternatives in order to support the analysis of the alternatives and decision making.

2.3.17 *Requirement: Follow Open Standards Where They Add Value*

Capgemini has always adopted open standards where they added value to the products that were being delivered. We have also contributed to an early architecture standard, IEEE 1003.23. This standard is now obsolete. It provided input for the famous IEEE 1471/2000 standard, which is commonly known and used in IT architecture. IEEE 1471 is now also ISO/IEC 42010. Currently Capgemini is member of The Open Group. We participate in the development of TOGAF, The Open Group Architecture Framework.

2.3.18 Requirement: Be Able to Effectively Demonstrate Completeness and Consistency

Common questions IT architects have to answer are: 'How do you know we have everything?' and 'How do you know everything is consistent?'. Many times our only answer was: 'You have provided me with the input. You know your business. Aren't you confident that we are complete and consistent?'. Of course this is the wrong answer. IT architects need to be able to demonstrate completeness and consistency. IAF needs to provide mechanisms that demonstrate it.

2.3.19 Requirement: Service Oriented Principles Have to Be Applied

Thinking in terms of service orientation has been adopted in very early stages within Capgemini. It clearly added value because it helped address a number of requirements.

Services have to be defined in such a way that they provide value to the service consumer. So the consumer has to be able to understand what the service will provide. This makes communication to the different stakeholders easier, and helps us to get decisions made.

Services encapsulate their internal structure and expose themselves through well defined, standardized interfaces. This helps manage complexity and thus supports one of the main reasons architecture has become a necessity in the IT industry – managing that complexity.

The quality of service that a service can deliver has to be well defined in order to the consumer to judge its value. Quality of service covers many non-functional aspects of the architecture.

Cost reduction is one of the ubiquitous requirements in our industry. Re-use of services is one of the things we use to reduce costs. Service orientation promotes re-use.

All the reasons above clarify why IAF needs to apply service oriented principles. Please note that we use the concept of a service not only in a technological way. We use this concept as well at business level, e.g. drilling a whole to find oil.

2.3.20 Requirement: Provide Support for Implementation Independent Models

It takes time to implement any architecture. The fact that it takes time will result in changes in the environment that could not be foreseen at the moment the architecture was created. We might want to take advantage of changes in the environment through time, simply because they might solve a problem we could not solve at the moment we created the architecture. One major example in this area is network bandwidth. Who could have imagined the speed we currently have 5 years ago? This is the reason to support 'implementation independent models'. These models show the structure of the architecture, but do not contain real life constraints. They can be used as a reference model at the time a certain part is being implemented.

2.3.21 Requirement: To Be Independent of, Yet Accommodating, Different Architecture Styles and Technology Innovations

As our industry is still rapidly evolving, different architecture styles are also evolving. Two tier client-server has been succeeded by 3-tier and cloud computing. Product oriented business organization is being replaced by customer oriented business organization. IAF has to be independent of these evolving architecture styles to be stable. On the other hand, IAF also has to be able to use the styles to create a specific architecture.

2.3.22 Requirement: Tool Independence

Different customers will have different tools to create and maintain architectures. We will have to be able to use the different tools in combination with IAF. We will not link ourselves to one tool environment. Capgemini has developed a meta model and certification scheme for tool vendors so they can embed IAF support in their tools.

2.3.23 Requirement: Diagramming Model Independence

IAF must not prescribe diagramming models. We need to be able to adapt to the diagramming models that are used in the customer's environment. We do however recommend that UML should be considered where appropriate because it is so widespread.

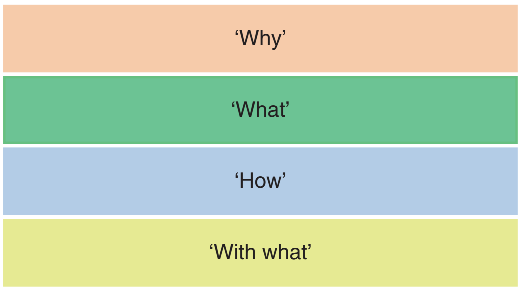
2.4 Logical Structure: The 'How' of IAF

2.4.1 Introduction

The previous section has described the requirements we have defined for IAF. Here we will describe the logical structure that has been created to fulfill the requirements.

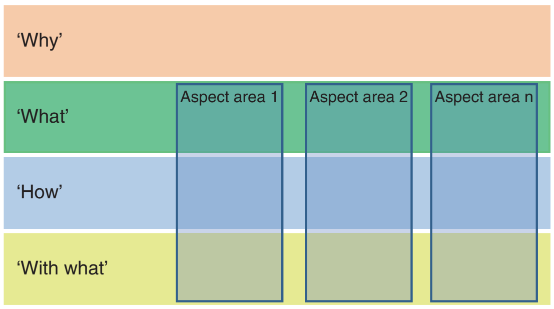
2.4.2 IAF content

2.4.2.1 Abstraction Levels



IAF recognizes abstraction levels. The abstraction levels are aimed at splitting one problem into smaller ones that are easier to solve. We follow the famous ‘Why, what, how, with what’ order in defining abstraction levels. First get the drivers, objectives, principles and scope right – the answer to the why question –, then understand the requirements – what services the solution has to support – , thirdly design how the ‘ideal’ solution will support the requirements, and finally decide with what physical components to implement the ideal solution. Abstraction levels need to be applied to all architecture topics, so they will be positioned horizontally across the topics.

2.4.2.2 Aspect Areas



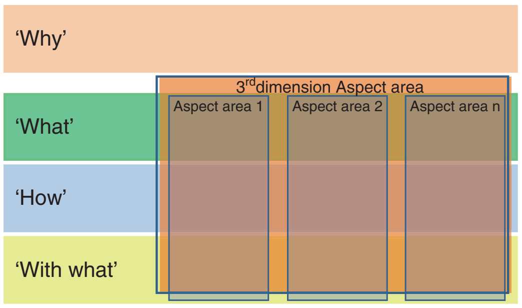
The Aspect Areas in the IAF describe a formal boundary between elements of the architecture solution that are usually considered within their own context. Each aspect area focuses on one particular dimension of the architecture, and adds information to the overall architecture. Commonly specific knowledge and background is required to be able to successfully address an aspect area. Aspect areas cover the what, how and with what abstraction levels. This is because the ‘why’ abstraction level contains observations and driving elements for the architecture such as strategies and trends that are applicable in all aspect areas.

Aspect areas are positioned vertically in the IAF diagram.

2.4.2.3 Third Dimension Aspect Areas

IAF also recognizes aspects that are fundamentally part of all other aspect areas, but often need to be addressed separately to ensure completeness and consistency.

Aspect areas in the third dimension commonly address topics regarding quality or non-functional aspects like security and governance, as these are the product of all aspect areas.



2.4.2.4 Artifacts

Artifacts are the core elements of IAF and fundamentally describe the architecture.

There are a number of core types of artifact within IAF that are essentially the same across any of the aspect areas in which they reside. This section describes these core artifacts. Other artifacts that are specific to an aspect area and abstraction level will be elaborated in Chap. 3.

Architecture Principles set out the general characteristics of the desired architecture and **why** it should be as it is. Principles are initially represented at the start of an architecture engagement; however they are often expanded and enumerated throughout the architecture process as architecture details are expanded, or as a result of better understanding of the business objectives.

Services are the architecture's fundamental building blocks. A service describes an 'element of behavior' or function needed in the architecture. The description of a service describes **what** it does, rather than *how* it is done. This implies that services are defined in the 'what' abstraction level.

Components are sets of services that are organized in accordance with the Architecture Principles and business objectives. The way IAF works with services and components is much different from many other architecture frameworks. See Sect. 2.4.2.10 for a detailed explanation. Components are defined in the 'how' and 'with what' abstraction level.

Collaboration contracts describe the interaction behavior between services and components. In effect they capture the non-functional aspects of the architecture. They document for example how often, how fast, how secure, and how controlled the interaction needs to take place.

Standards are documented statements that describe what has to be adhered to during the realization of the architecture. We often distinguish two types of standards, based on the moment they have to be adhered to. If a standard can be adhered to in the next change of the system, then it is a normal standard, and can be treated as described. If adherence to the statement has to be realized before a certain date, like with law changes, then we use the term 'rule'. Commonly standards and rules are non-negotiable. Senior business management needs to decide if they can be breached.

Guidelines provide guidance and direction (requirements) for the realization of the architecture. They should normally be adhered to. Commonly specific procedures are put in place to manage adherence to guidelines. One has to obtain waivers if it’s not possible to adhere to guidelines.

Specifications describe how specific architecture components should be built, configured and implemented.

2.4.2.5 Viewpoints and Views

Views are a structured organization of the architecture artifacts in accordance with a given criteria. Views are primarily the constructs for representing the architecture (usually of structure) from different perspectives or viewpoints.

A view is the representation of an artifact or the combination of artifacts from one or more aspect areas with a specific objective. A view is a very flexible element of IAF. Depending on the architecture engagement an architect will create different views, each providing different insights. Views are very effective as a means for communicating the architecture. They are also a critical tool when analyzing the problem; by looking at the problem from a specific viewpoint we can identify areas of concern, look for gaps, etc.

Interaction Models and *Cross-references* are two fundamental Views used in IAF to show the basic architecture structure and relationships. Interaction Models typically describe the relationships between similar artifacts within a specific Aspect Area and Cross-references typically describe relationships between artifacts across different Aspect Areas. Cross-references are one of the key mechanisms for traceability and decision justification in IAF.

Other Views are selected as required usually driven by the Architecture Scope and Objectives. Views are therefore something that the architect selects based on need of the stakeholder and as such there is no definitive list of Views within the IAF.

Some views however are regularly used and are instrumental in describing significant relationships within the IAF.

IEEE 1471[1] uses the following descriptions for view and viewpoint:

A view is a collection of models that represents the whole system with respect to a set of related concerns. A view belongs to a particular architectural description. For example, a structural view of a system might include a model showing components, their interfaces and the classes comprising them, and a model of their dependencies and inheritance relationships. A performance view might consist of models for resource utilization, timing schedules and cause-effect diagrams. We use terms like

[1] IEEE Computer Society (1999) IEEE P1471 Recommended Practice for Architectural Description. IEEE, US.

'operational view' and 'performance view' where others have used terms like 'operational architecture' and 'performance architecture.'

A viewpoint captures the rules for constructing and analyzing a particular kind of view. It is a template for a view which can be reused across many architectural descriptions. The term 'view type' was considered as an alternative for viewpoint because of the strong analogy of view and viewpoint to instance and type; but we chose 'viewpoint' because of its use in existing standards and the requirements engineering literature.

TOGAF version 8[2] and 9[3] use the following definitions:

View: A 'view' is a representation of a whole system from the perspective of a related set of concerns. A view is what is seen from a viewpoint. An architecture view may be represented by a model to demonstrate to stakeholders their areas of interest in the architecture. A view does not have to be visual or graphical in nature.

In capturing or representing the design of a system architecture, the architect will typically create one or more architecture models, possibly using different tools. A view will comprise selected parts of one or more models, chosen so as to demonstrate to a particular stakeholder or group of stakeholders that their concerns are being adequately addressed in the design of the system architecture.

Viewpoint: A definition of the perspective from which a view is taken. It is a specification of the conventions for constructing and using a view (often by means of an appropriate schema or template). A view is what you see; a viewpoint is where you are looking from – the vantage point or perspective that determines what you see.

IAF uses views and viewpoints in the same way that IEEE 1471 and TOGAF do. Often we define the stakeholder and the concern the stakeholder has along with the description of the view. We do not prescribe modeling or diagramming techniques in the viewpoints as that would be in contradiction with our requirement regarding diagramming model independence.

2.4.2.6 Solution Alternatives

It is most common that a single solution does not exist that will meet all stakeholders requirements. IAF supports a technique to investigate different solution alternatives and to discuss these with the stakeholders. The place where this should be considered is in the abstraction levels 'How' and 'With what' because these levels are the places where decisions are made regarding the structural elements of the architecture. Commonly solution alternatives are defined per aspect area, especially at the logical level. This is done to simplify the analysis of the different alternatives. Of course the solution alternatives per

[2] The Open Group (2007) TOGAF Version 8.1.1 Enterpise Edition. The Open Group, US.

[3] The Open Group (2009) TOGAF Version 9. The Open Group, US.

aspect area can be merged into one overall analysis of the solution alternatives for the whole architecture. There are two basic approaches to solution alternatives, the 'fast track' and the 'full analysis' approach.

Within each approach you need to define the criteria that are used to compare the different alternatives. After that the different alternatives need to be identified. It is best practice to base alternatives on architecture principles that have been defined. That enables you to rationalize the identification of the solution alternative, and the scoring against the defined criteria and principles.

The fast track approach scores each solution alternative against each criterion. The amount in which the alternative fulfills the criterion determines the score. The solution alternative that fits all criteria the best, wins.

The full analysis approach enables the usage of relative weights for the different criteria, thus making some criteria more important than others. The table below shows how it works.

Solution	Criterion 1			Criterion 2			Criterion *n*			End result
alternative	Weight	Score	Result	Weight	Score	Result	Weight	Score	Result	
Alternative 1	3	3	**9**	2	3	**6**	1	1	**1**	***16***
Alternative 2	3	2	**6**	1	3	**3**	1	1	**1**	***10***

This approach takes more time because you do not only have to agree the criteria and scores, but also the weight of each criterion.

Solution alternatives obviously should also take the different interests different stakeholders have into account. For example, centralization might be in the interest of the corporate staff departments, whilst being strongly opposed by business unit management because centralization implies loss of control for them.

2.4.2.7 Domains

Architectures can become relatively large, simply because they have to describe many services and components. An enterprise level architecture can easily contain hundreds of services and components. It is especially difficult to communicate and visualize services, as they are the fundamental building blocks of the architecture, and therefore the most abundant. Architects using IAF have implicitly solved the communication and visualization challenges they had with services by grouping them together in ways the stakeholders can relate with. They often used the term 'domain' or 'segment' to describe the groups. As of IAF 4.5 we have formalized the usage of domains, especially in the 'what' abstraction level, as that is where services are defined. There has been much debate regarding this subject, as many people argue that grouping services into domains implies creating a structure, and thus is part of the 'how' abstraction level. There is a fundamental difference between the way we use domains and the way we construct components.

Domains are based on things we want to communicate and stakeholders can relate with. Common examples of domains are business units or geographical locations. The intention of domains is to visualize and communicate services so they can be validated by stakeholders.

Components are constructed on the basis of architecture principles, which can be very different from the basis used for domains. An example is architecture principle 'buy before build'. This implies that we should group our services in such a way that they reflect what we want to have in a package we are potentially going to select.

2.4.2.8 Synonyms

For a long time we did not have a formal mechanism to document the terms we use to communicate artifacts to non-architect stakeholders. This has lead to situations in which architecture teams caused confusion due to incorrect usage of terminology. As of IAF version 4.5 we have formally introduced the usage of synonyms. The IAF glossary has been extended to allow the definition and usage of synonyms. This can be done on a per project or per client basis. Another advantage of synonyms is that it makes it easier to link to terminology that is already being used in the organization.

2.4.2.9 Mechanisms

Wikipedia[4] describes the term 'mechanism' like this:

*A **mechanism** is some technical aspect of a larger process or mechanical device, or combination of parts designed to perform a particular function. Sometimes an entire machine may be referred to as a mechanism. Examples are the steering mechanism in a car, or the winding mechanism of a wristwatch.*

The term 'mechanism' was used in IAF version 1 and 2 to describe parts of the overall architecture that provided a distinct function. Often they consisted of combinations of artifacts from multiple aspect areas. The mechanisms were even described in mechanism catalogues. The usage of the concept mechanism has been less prominent in the current versions of IAF. This does not mean that they cannot be used.

Common and current usage of mechanisms is often within quality related areas. Mechanisms that ensure specific quality aspects can be described and even re-used in different parts of the architecture. Examples of mechanisms that could be described are a high availability mechanism that prescribes the different services and components that need to be used to achieve a certain level of availability, like synchronous replication and hot standby. Another example could be the use of biometrics mechanism for implementing strong authentication as opposed to weak authentication based on the combination user-id + password.

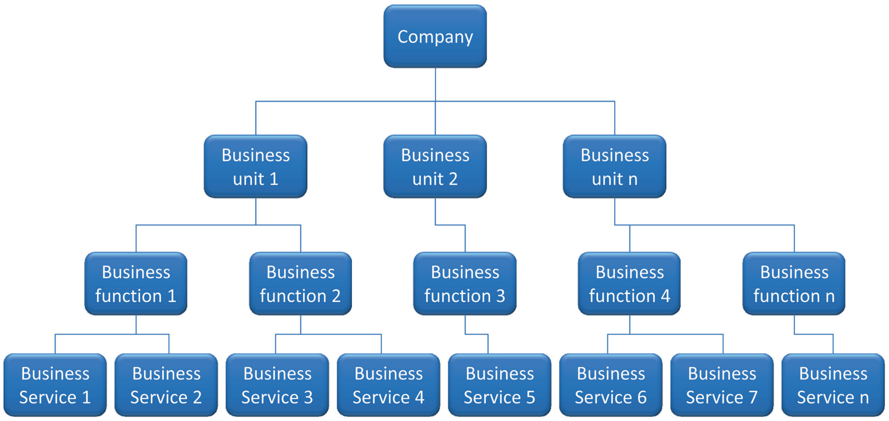
[4] Information available via Wikipedia. Http://en.wikipedia.org. Accessed December 2008.

2.4.2.10 Creating Components

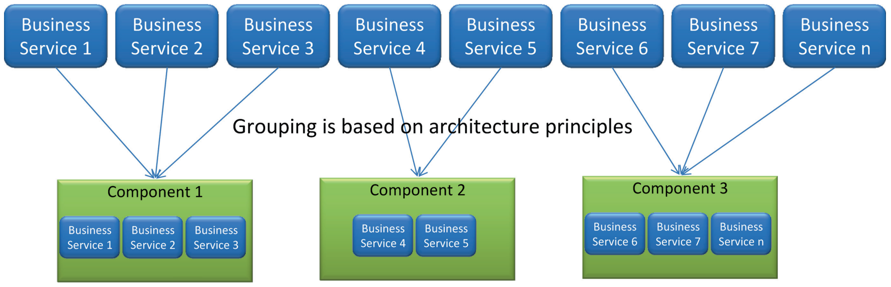
Many architecture frameworks take a top down approach to the creation of the architecture. They start high level, and decompose down to the level required to document what the architecture consists of, much like in the figure below. This approach has advantages and drawbacks.

The main advantage is that stakeholders can easily relate to the structures and therefore easily validate them.

The main drawbacks of this approach are: (1) It’s not that easy to define and analyze solution alternatives, as thinking is guided and influenced by the hierarchical structure that is being created. The structure makes it harder to think out of the box. (2) The architect is implicitly combining the ‘what’ and the ‘how’ question, because he is defining the structure along with the definition of the requirements. This contradicts the requirements we intend to meet within IAF.



IAF takes a different approach. First we define the services at the level of detail required without explicitly putting them into a hierarchical structure. Then we define grouping criteria that are based on the architecture principles. After that we create components by grouping the services into components based on the grouping criteria. The figure below visualizes the IAF approach.



Advantages of the IAF approach are that it is easy to define and create solution alternatives as we are not influenced by structures already created. We also explicitly split the 'what' from the 'how' question by first defining services and then grouping them. Of course there is a drawback to this approach: visualization and communication of the services is more complex. This has been solved by the introduction of domains.

A group of services is a component for which a solution as a whole exists, and that can function as a whole. In general each architecture will only contain one group for each required set of services.

2.4.2.11 Policy and Collaboration Contracts

To create an architecture that works, there has to be a balance between supply and demand. In other words, the services have to be able to supply what the service consumers demand from them. An example will clarify the importance: If a pizza restaurant can bake 10 pizza's a time, and it receives one customer that orders 8 pizzas, all is fine. If a second customer comes in and orders 5 pizza's there is a problem. To be able to balance supply and demand we need to document both of them. As stated in Sect. 2.4.2.4, the collaboration contract is used to document the interactions between services and components. This implies that they document the non-functional attributes of what the service consumer demands. To be theoretically correct there would have to be a second type of contract to document the supply side, a so called 'policy contract[5]'. Services and components could have multiple policy contracts to document their different capabilities (e.g. a service request that requires immediate response, and one that can be deferred). This would lead to a jungle of services, components, collaboration contracts and policy contracts. To simplify things, IAF has decided to incorporate the attributes that describe the policy contract into the attributes of the services and components. If a specific architecture needs to, they can remove the attributes from the services and components, and introduce policy contract artifacts.

2.4.3 IAF Process: Engagement Roadmaps

IAF deliberately separates the process of architecting from architecture content, because the content is relatively stable and the process by which the engagement is run, will be different for each engagement. The process depends

[5] This term was chosen to align with the open standard WS-Policy, which describes how the capabilities of a web service can be documented and published.

on the context of the engagement: time frames, stakeholders, organizational culture, the architects team, etc will differ by situation.

We have defined the term **engagement roadmap** as '*a process pattern describing how to run an architectural engagement for specific architectural objectives within a specific client. It specifies architecture content as well as the engagement process'*. The concept of roadmap allows us to apply the line of thought from IAF in various sequences, depending on the type of assignment. Since IAF provides an artifact framework, it will provide consistency in the registration of and reasoning behind deliverables. The term 'Engagement Roadmap' is used explicitly to differentiate the process of architecture from Architecture, Product or Technology Roadmaps which describe the evolution of the architecture, product or technology.

Ideally Roadmaps are documented in the same way as patterns are. That facilitates re-use. Roadmap descriptions should have the following attributes:

Name	*[Identify the roadmap]*
Version & date	
Description	*[Provide an overview of the roadmap – an introduction.]*
Context	*[Describe the context that led to the definition of the roadmap.]*
Architecture areas covered	*[Identify which aspect areas and abstraction levels are covered]*
Design decisions & rationales	*[Provide insight into how and why the roadmap was constructed]*
Pre and post conditions	*[Help the user understand when the roadmap can be used, and what the effect of its use will be.]*
Open issues	*[Make the reader aware of things that have not been addressed in the roadmap.]*
Potential pitfalls	*[Make the user aware of potential risks or problems associated with the use of this roadmap.]*
Newly created problems	*[Make the reader aware of additional/new things that have to be addressed as a consequence using this roadmap.]*
Contacts	*[Provide information on where to get more details about the roadmap.]*
See also links	*[Links to related roadmaps, patterns and case studies that have applied this roadmap.]*
Roadmap details	*[The detailed description of the roadmap]*

A set of commonly used roadmaps can be found in Sect. 5.5.

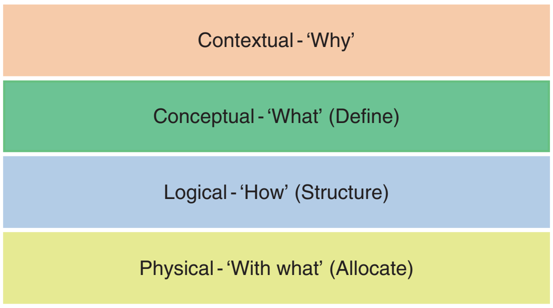
2.5 Physical Elements: The 'With What' of IAF

2.5.1 Introduction

In Sect. 2.2 we addressed the 'why' question in regard to IAF. Sect. 2.3 describes IAF's requirements and thereby addresses the 'what' question. The 'how' question has been addressed in Sect. 2.4 by describing IAF's logical elements. This section uses the logical elements to describe which real life physical elements are part of IAF. An example: We described that we use abstraction levels in Sect. 2.4.2. In Sect. 2.5 we will describe which physical abstraction levels are really there.

2.5.2 Physical Content

2.5.2.1 Abstraction Levels


Contextual - 'Why'
Conceptual - 'What' (Define)
Logical - 'How' (Structure)
Physical - 'With what' (Allocate)

The real life abstraction levels that have been defined in IAF are fairly obvious. We have been using them throughout this chapter. The names for the abstraction levels were adopted from a large architecture project Capgemini had executed at the Bibliothèque nationale de France in Paris. The project was about implementing distributed computing in the French library. There they captured requirements in terms of 'conceptual servers'. They translated the requirements into the logical structure of the computing environment, and called the different components 'Logical servers'. And of course the next one is easy to guess – The real life servers were called physical servers.

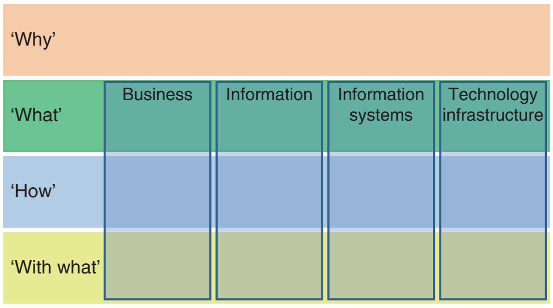
Through time we also introduced synonyms for the terms 'What', 'How' and 'With what'. The diagram maps the interrogative pronouns with the more formal IAF name of the abstraction layers: Contextual, Conceptual, Logical and Physical. In effect the 'what' level is all about *defining* architecture requirements. The 'how' level actually creates the logical *structure* of the architecture. In the 'with what' level the logical components are *allocated* to real life, physical things you can buy, hire or build.

In the past there has been debate about defining a fifth level, which would address the 'when' question. The 'when' question is all about transformation and migration planning – when will we transform or migrate which part of the architecture. As transformation and migration planning is not the sole responsibility of the architect, and as the planning depends on many other aspects, it has not been incorporated as a formal part of the IAF model.

2.5.2.2 Aspect Areas

IAF consists of four physical aspect areas, Business, Information, Information systems and Technology infrastructure.

There are a number of rationales behind the choice of these aspect areas. The rationales are based upon 2 questions: (1) What do I need to know to deliver the overall architecture in such a way that it serves its purpose and (2) Which capabilities are required to answer the different types of questions we need to address in the architecture.



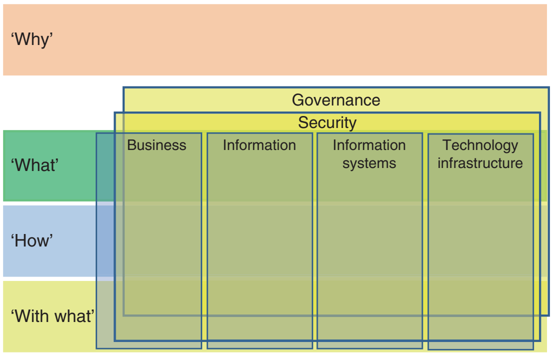
To answer the first question: IAF was originally designed to support the creation of architectures for Capgemini's business. Capgemini's business is all about business and technology transformation. So the topics we need to address are:

1. The structure of the business itself, otherwise we will not be able to understand which technology is needed to support or enable the business;
2. The way in which the business wants to process its information, as information processing is what the supporting technology does;
3. The structure of the information systems that support the business, to be able to understand which information systems have to be built and bought, along with the interfaces between them;
4. The computing systems, network technology and other infrastructural elements needed to make the information systems work.

The second question is actually implicitly answered by the first question. The capabilities a person needs to be able to structure a business are different from those needed to structure the information household of the business. In the first instance one needs to know the pros and cons of different business models and the way processes are structured within the different models. People that need to structure the information household of a company need to know what information modeling is all about. An information modeler does not have the skills that information systems people have. Information systems architects need to know the different packages that are available for a specific area, they also need to know all about interfacing mechanisms, data conversion and migration topics etc. People with information systems skills do not naturally have the capabilities to define and structure the computing systems and network technology needed to let the information systems run.

This all leads to the conclusion that the four aspect areas defined in IAF are needed to create an architecture that supports business and technology transformation, and that each area requires distinct skills which helps in getting the right architect in the right place, enabling effective communication across the borders of each architect's discipline.

2.5.2.3 Third Dimension Aspect Areas



Next to the four aspect areas covered so far, IAF has included two third dimension aspect areas, Security and Governance in previous versions. Security is a topic that must be addressed holistically, across aspect areas, to ensure that all security related topics in the different areas work together to provide the desired level of security. You can enforce very tight security within an information system, but if you do not ensure that access control to buildings in which the information system is used is also at the desired level, it will be relatively easy for a hacker to enter the building and access the system through the workplace of one of the employees that has left his desk. Security architecture is also a topic that requires specific knowledge, so it also fits the rationales of the aspect area definition.

Governance, the other third dimension aspect area, has a bit of a strange name, especially if you look at what topics it addresses. Capgemini still considers the options to cover many more topics within this aspect area, including subjects like compliance or corporate governance. We have started with addressing quality of service (QoS) aspects within this area. Thus, the aspect area "governance" is all about ensuring that the desired quality of service is delivered at the defined acceptable level of cost. Of course quality of service is not only delivered by IT. There are many processes around IT that have their own information processing requirements and information systems. Simply think of topics like incident management, change management and availability management.

During the creation of IAF version 4 we have merged Security and Governance into the main aspect areas. The rationale for this was the observation that security and governance were often neglected as topics, were only addressed at the end of the process and then required the collection of additional information from the business, and at this point were often looked at only from an IT perspective. People were only focusing on the main aspect areas. The merge was done by the addition of attributes to artifacts in the main aspect areas and defining mandatory security and governance views in the relevant cells of IAF. This has the effect of both ensuring that the information can be collected from the business as early as possible, relates directly to the business needs and is treated in a holistic manner across the whole architecture.

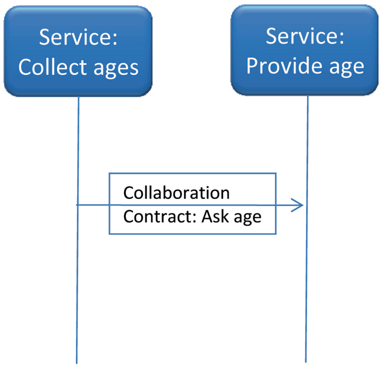
2.5.2.4 Artifacts

As Chap. 3 will describe all artifacts per aspect area and abstraction level, we will use this section to explain where artifact types are positioned in the IAF.

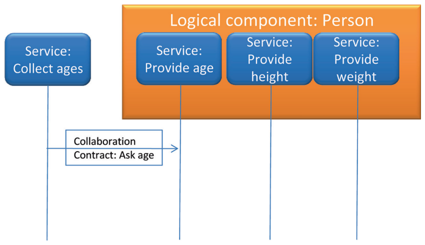
The contextual level is the area where all our 'input' is positioned. The input help us understand (1) the context of the business (mission, strategy, drivers, . . .), (2) the architecture engagement (scope, objective, . . .) and (3) the architecture principles.

Services and collaboration contracts are used in the conceptual level to document architecture requirements. What services are needed and how do they collaborate? Services are defined within all aspect areas, so they can be business services or technology services.

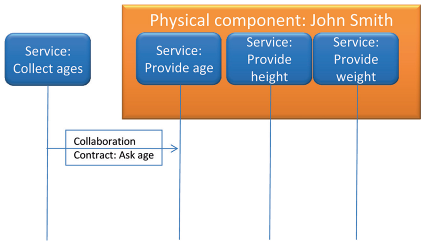
Actually this way of documentation can be used to describe functional and non-functional aspects of anything. The service describes what can be delivered (functional aspects) and how it should behave (non-functional aspects). The collaboration contract describes the way in which it is to be delivered in terms of the communication mechanism to be used and the syntax and semantics of how it can be requested. The communication mechanism effectively defines how fast, how secure, how often etc things can happen. An example will clarify this. If you were in a conversation and you were asked what your age is, then nine out of ten times you would answer the question within a few seconds. This is an example of two services executing a collaboration contract with each other. The requested and delivered service are: 'Collect age' and 'Provide age'. The communication mechanism is sound waves generated by vocal chords. Syntax and semantics are the English language. The communication mechanism implicitly defines behavior: you know the other expects an answer within a few seconds. You also know you can lie about your age, and the



other might not mind. However, if we used another communication mechanism, your reaction would be different. If the question was asked in an email, you might answer it. You have the choice not to. You can also wait a week before you answer. The requestor knows that. And you can still lie about your age.



Components are positioned in the logical and physical levels within IAF. They are sets of one or more services, represented either in a logical or physical manner. Each main aspect area contains components. Components also have collaboration contracts between them, as they are sets of one or more services and services have collaboration contracts.

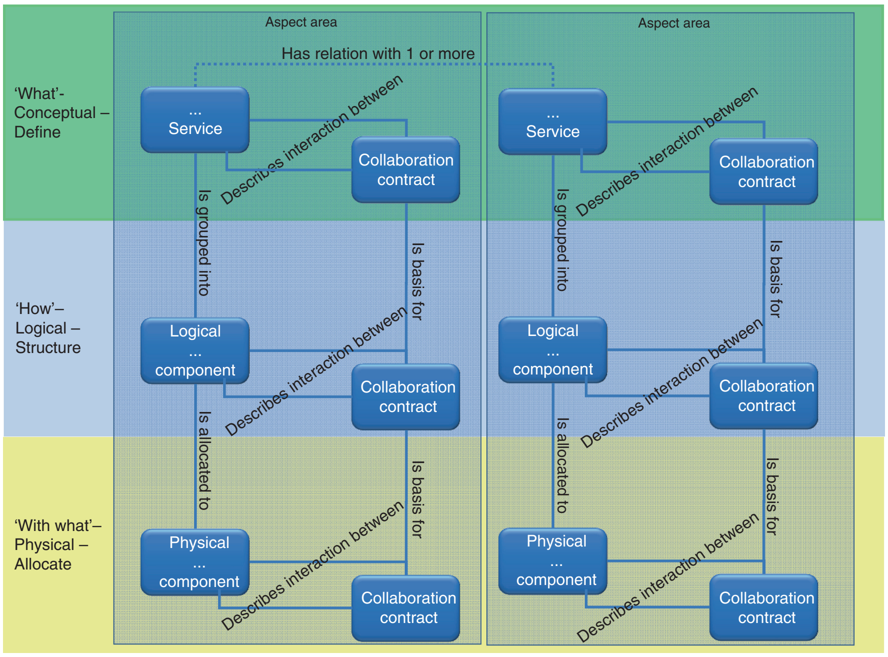


Standards, guidelines and specifications can be relevant in every aspect area. As they are key deliverables that are to be used to guide realization of the architecture, they are positioned in the physical level. So IAF identifies business standards, information standards, information systems standards and technology infrastructure standards. Where relevant you can also define security and governance standards. Guidelines and specifications are treated in the same way.

2.6 Recap: IAF's Meta-meta Model

This paragraph brings several topics from the previous paragraphs together to show how they are linked. The meta-meta model depicted in the figure visualizes the relationships. Services are defined in the conceptual level. Interaction between services is documented in collaboration contracts. Services in different aspect areas will have relations with each other. These relationships are commonly documented in cross-references. Services do not need to have one-to-one relationships with services in other aspect areas. It is not uncommon to encounter one-to-many and even many-to-many relationships.

Services are grouped into logical components. Grouping is based on principles. Grouping always implies trade-offs, which we assess using the architecture principles. Different groupings can be made and compared with each other. The name we use for a set of groupings is a **solution alternative**.


Aspect area
Aspect area
Has relation with 1 or more
'What'- Conceptual – Define
'How'– Logical – Structure
'With what'– Physical – Allocate
... Service
Collaboration contract
Describes interaction between
Is grouped into
Is basis for
Logical ... component
Is allocated to
Physical ... component

The collaboration contracts between services are the basis for collaboration contracts between components. Collaboration contracts between components can be the result of merged collaboration contracts between services, if their characteristics allow them to be merged. If they cannot be merged, then we can conclude that there will be two 'interface types' between the components. A simple example is one interface for single, high priority service requests, e.g. airline ticket bookings. The second interface type could then be grouped requests for lists of bookings done during the last 24 hours. Collaboration contracts in one area also form the basis for the collaboration contracts in other areas. A business area collaboration contract could define the maximum business response time for a service request to be 3 seconds. We can use that as the basis for the response time for the IS services that support the business service. We could assume that IS processing would take 2 seconds. That implies that the maximum infrastructure response time would have to be 1 second.

Logical components are 'allocated' to physical components. What does this 'allocation' actually mean? Well, in its essence the mechanism is simple. Logical processes are mapped to the real life physical parts of the organization that will be executing the processes. Logical IS components are allocated to products that have been selected to implement them, or we have decided to build them in a specific physical environment like .NET or Java. Logical TI components like a server, mobile workstation, SAN or backbone switch have been allocated to the real life physical products with which they will be implemented.

This basic approach and meta-meta model has proven to work in all architecture aspect areas. It keeps the overall architecture as simple as possible and ensures traceability across aspect areas.