Chapter 4
The Results of Enterprise Architecting

4.1 Introduction

During the process of enterprise architecting several results can be produced. These results are not limited to principles, models (including their cross-references) and views alone. Other results are for example intermediate results used to develop the enterprise architecture and the evaluation of alternative solutions/directions. Some of the important results of an enterprise architecting effort do not even have to be tangible, for example shared understanding, shared agreement, and commitment amongst stakeholders.

The actual set of results that should be produced, and the required level of detail and form of the results, depends on the stakeholders and their concerns, as well as the decisions that should be taken based on these results. Even in a simple example it becomes clear that enterprise architecture can produce a large variety of tangible and intangible results. Therefore criteria on product quality are needed to make choices which results should be delivered. In such choices we can also profit from an insight how the possible results interrelate, as described in architecture frameworks.

The remainder of this Chapter is structured as follows. As a simple basis for the whole Chapter, we will start providing examples of enterprise architecture results, using the relatively simple case enterprise: *Pizzeria "Perla del Nord"*. Then we will reflect on product quality and its criteria: when is good *good enough*? Subsequently we will revisit architecture frameworks as a means to structuring tangible enterprise architecture results. This is followed by a discussion of a methodical perspective on the creation of models and views used in enterprise architecture. Before concluding, we will also argue the point for a language with a unified look and feel for the representation of architectural models.

4.2 Example enterprise architecture: Pizzeria "Perla del Nord"

In the remainder of this Chapter we want to be able to regularly refer to specific enterprise architecture results. We therefore introduce a running example: *Pizzeria "Perla del Nord"*. This running example is primarily inspired by [34], while the principles used in the example are derived from [36]. It should be noted that the Pizzeria case is a small case, and is not intended as being exemplary for enterprise architecture cases in real life. The purpose of the Pizzeria case, however, is to illustrate the workings of some key elements in enterprise architecture, and not to mimic a real-life enterprise architecture. The latter is beyond the goals of this introductory book.

For *Pizzeria "Perla del Nord",*we will discuss its current objectives, its design principles and some models reporting on its current design. We meet the pizzeria just at the moment they intend to play a significant role in the business-to-business (B2B-)market. This is the result of a change of strategy, in the sense that the pizzeria intends to open up a new market (business-to-business). For this intention we will subsequently show the impact-of-change, as required by the CEO of Perla del Nord to govern the intended transformation. As the CEO's desires are the primary driver for the creation of this architecture, the focus of the example will be on the business aspects.

4.2.1 Current situation

The pizzeria is located in the city centre of a medium sized town. In and around this city centre a large number of offices are located. The mission of the pizzeria is *to offer positive influence in the work-life balance of both yuppies and dinkies*. The desired future state is *to be the most renowned pizzeria in the region* and *be the number one customer preference* which will be accomplished by *organic growth* and *sharing the profit with employees*. This vision is made more specific in terms of a number of goals (*grow from 100 pizzas a day to 150 pizzas per day in the next three years, improve profitability by 10% per year and to improve quality, measured by diminishing customer complaints*) and policies (*management development* and *financing programme for talented employees to support them in starting their own branch*).

In the current situation, the pizzeria is designed according to a number of business principles:

1. *Reward valued customers*
2. *Outside own organisation: get money first*
3. *Bake to order (not to stock)*
4. *Bake while driving*

No explicit technical infrastructure principles have been used in the current organisational design of the pizzeria. In discussing the pizzeria case, we will consider the pizzeria at three levels of abstraction (leading to a clear separation of concerns):

Valuation – The value exchange between the pizzeria and its environment, focusing on the value each actor potentially adds to the (economic) goals of other actors.

Function – The functionality provided by the pizzeria to its environment in terms of business services, transactions and their orchestration. In the case of the pizzeria, the business services will be services that are offered to clients, in line with the business goals identified in the pizzeria's strategy.

Construction – The construction of the pizzeria in terms of sub-actors.

The distinction between these levels is motivated as follows:

- Depending on the actors, and their goals, a business service (such as *deliver pizza*) may provide different values to different actors. This motivates the distinction between the valuation and function layers.
- A given business service may be realised using different organisational structures. In other words, for a given business service, several alternative constructions can be used for their realisation. This provides the motivation for a distinction between the function and construction layers.

In modelling the current design of the pizzeria at these three levels, we use three different notational styles:

1. e3Value [46],
2. DEMO [114] and
3. ArchiMate [78].

The highest-level view of the pizzeria, the *valuation* level, is shown in Figure 4.1 in terms of an e3-value diagram. *Customers* and the *Pizzeria* may engage in a value exchange where money flows to the *Pizzeria* in exchange for a pizza. *Customers* are regarded as the initiators of this value exchange.

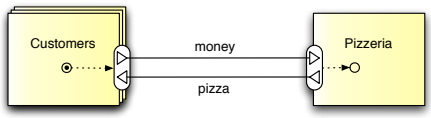

Fig. 4.1 Pizzeria at the valuation level

At the next level of abstraction, we are interested in the functionality provided by the Pizzeria to its environment. For this situation we assume that the money/pizza value exchange is to be realized in terms of two business services, namely *complete purchase* and *pay purchase*. Below we will see how moving to a B2B context will lead to a third business service. The diagram depicted in Figure 4.2 is a DEMO construction model, expressing the coherence (chain/network) of business services,

delivered by actors to other actors within a defined scope. In DEMO each business service is delivered by a transaction, consisting of a production embedded in four standard phases of co-ordination, as follows:

Request – An actor requests a business service from another actor.
Promise – The requested actor promises to deliver the business service.
Now the requested actor produces the requested result, i.e. the service is executed.
State – The requested actor states/claims it has indeed delivered the service.
Accept – The requesting actor accepts the result of the business service.

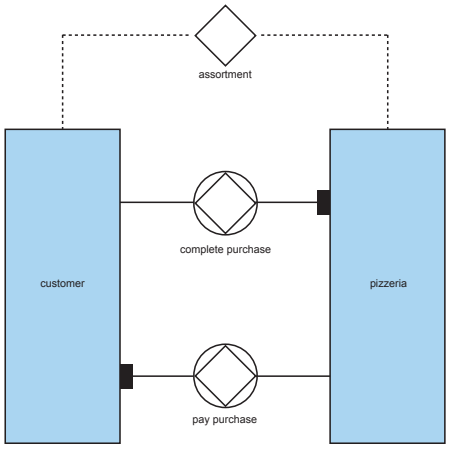

Fig. 4.2 Transactional view of value exchange

The transaction symbol combines a circle, signifying the four coordination phases, and a 45° rotated square, signifying the production. DEMO also discerns several exception handling procedures since there are likely to be numerous ways in which a transaction can break down.

When considering the two transactions between a customer and the pizzeria, which are involved in a pizza-for-money value exchange, there are several ways to interleave the four phases of the transactions. Should a pizza be paid before it is delivered? Or even before the pizzeria promises it to be produced? A selection between these options should ultimately be motivated in terms of the pizzeria's strategy.

In Figure 4.3 we have depicted the current orchestration of the two transactions. In this orchestration, the *customer* places an order (e.g. by phone) and immediately pays for it (e.g. using a credit card). The pizzeria responds to the request for purchase by a request for payment, which is required to be followed by the actual payment (promise / state / accept), after which the pizzeria promises to deliver the pizza (and indeed will do so).

When we decompose the construction of the pizzeria, this leads to the top-level construction as shown in Figure 4.4. We now see the chain of dependencies within

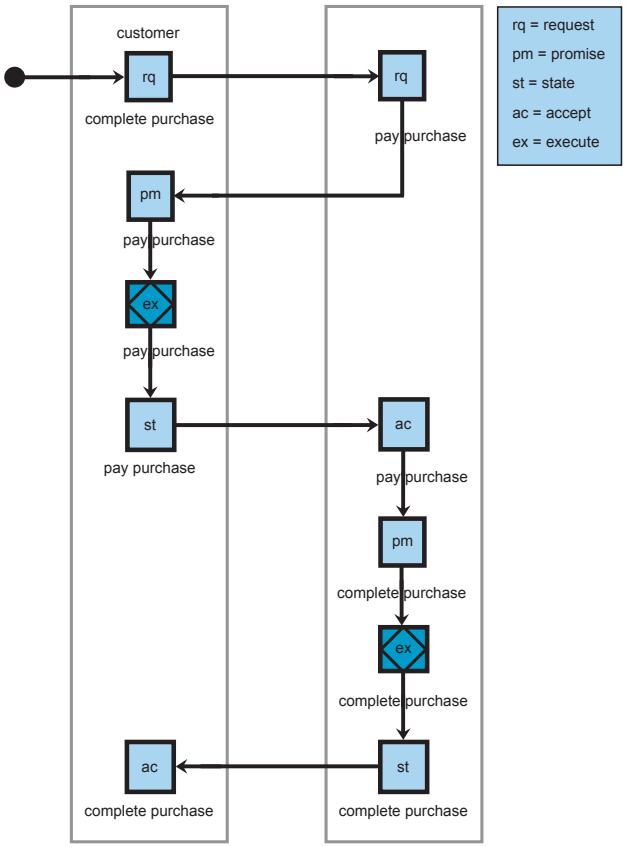

Fig. 4.3 Orchestration of transactions

the pizzeria: it is actually the *completer* who provides the business service *complete purchase* and uses the business service *pay purchase*. What is not shown in this diagram, is the fact that in executing *pay purchase*, the *completer* will offer customers a *discount* based on their order history. The *completer* in turn depends on the *baker* for the *baking of the purchase* and a *deliverer* for physically *delivering the purchase*. Also extra business resources appear now: the baker of course refers to the same assortment, but also uses *recipes*, the completer uses *customer data* and the *deliverer* uses *maps*.

The decomposition of the pizzeria’s construction also allows for a refinement of the transaction orchestration. This is shown in Figure 4.5. After the payment by the customer has been executed, the *completer* confirms the order to the *customer* and orders the *baker* and the *deliverer* to bake and deliver the order to the *customer*. Baking and delivering the order are executed in parallel: the *baker* prepares the pizza and places it in the oven on the back of the moped. Finally the order is delivered to the *customer*.

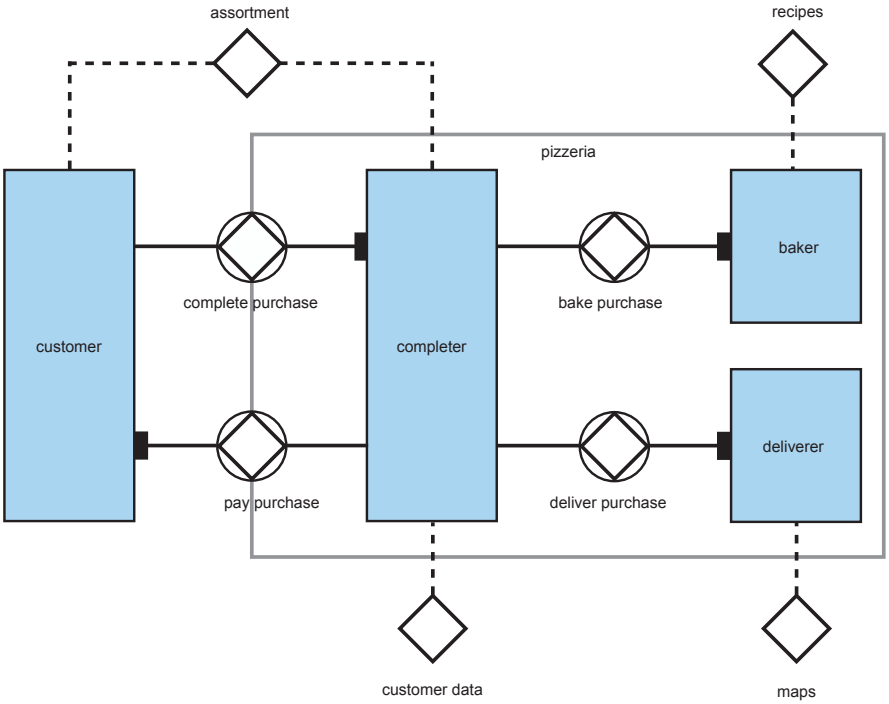

Fig. 4.4 Decomposition of the Pizzeria

Looking at the current situation in terms of principles, we see that the principles have been implemented as follows:

- *Reward valued customers.* Returning customers get a discount based on their purchase history.
- *Outside own organisation: get money first.* The order has to be paid before it is confirmed.
- *Bake to order (not to stock).* The order is baked after the customer has ordered.
- *Bake while driving.* Baking the order and delivering the order are executed in parallel by using mobile baking technology.

Figure 4.6 provides an overview of the orchestration of transactions including actor roles responsible for each phase in the transaction. Actor roles are implemented by *functionary types*, being a cluster of responsibilities for co-ordination and production acts, such that a person can be assigned to that cluster.

As a general rule, the functionary type who performs the promise in a transaction is considered to be the one who is authorized to be the executor of all phases of that transaction. But due to obvious reasons, the functionary that promises the transaction *complete purchase* (order taker) will not physically go to each *customer* to state the *complete purchase* transaction. This authority is, in our case, delegated

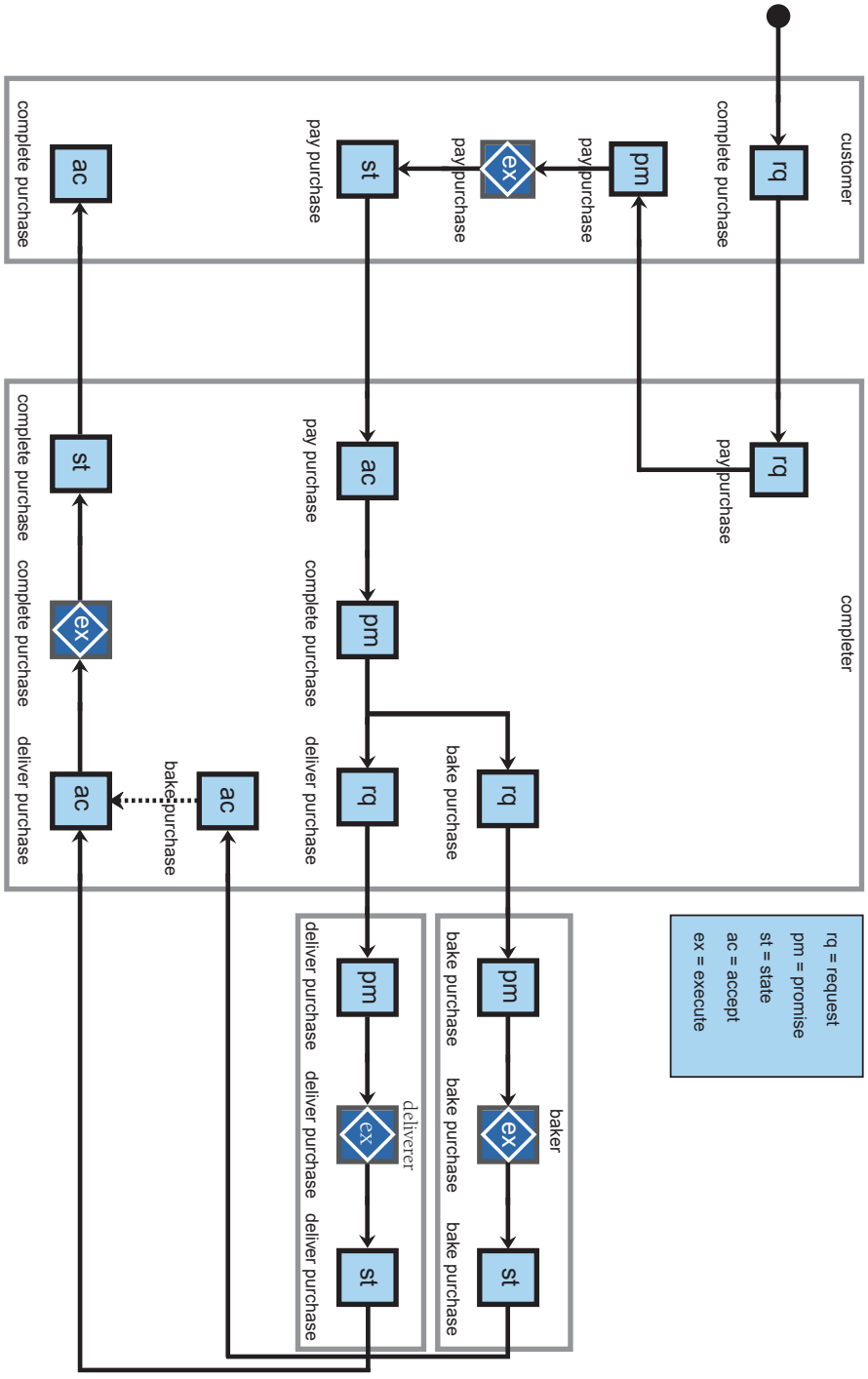

Fig. 4.5 Composed transaction orchestration

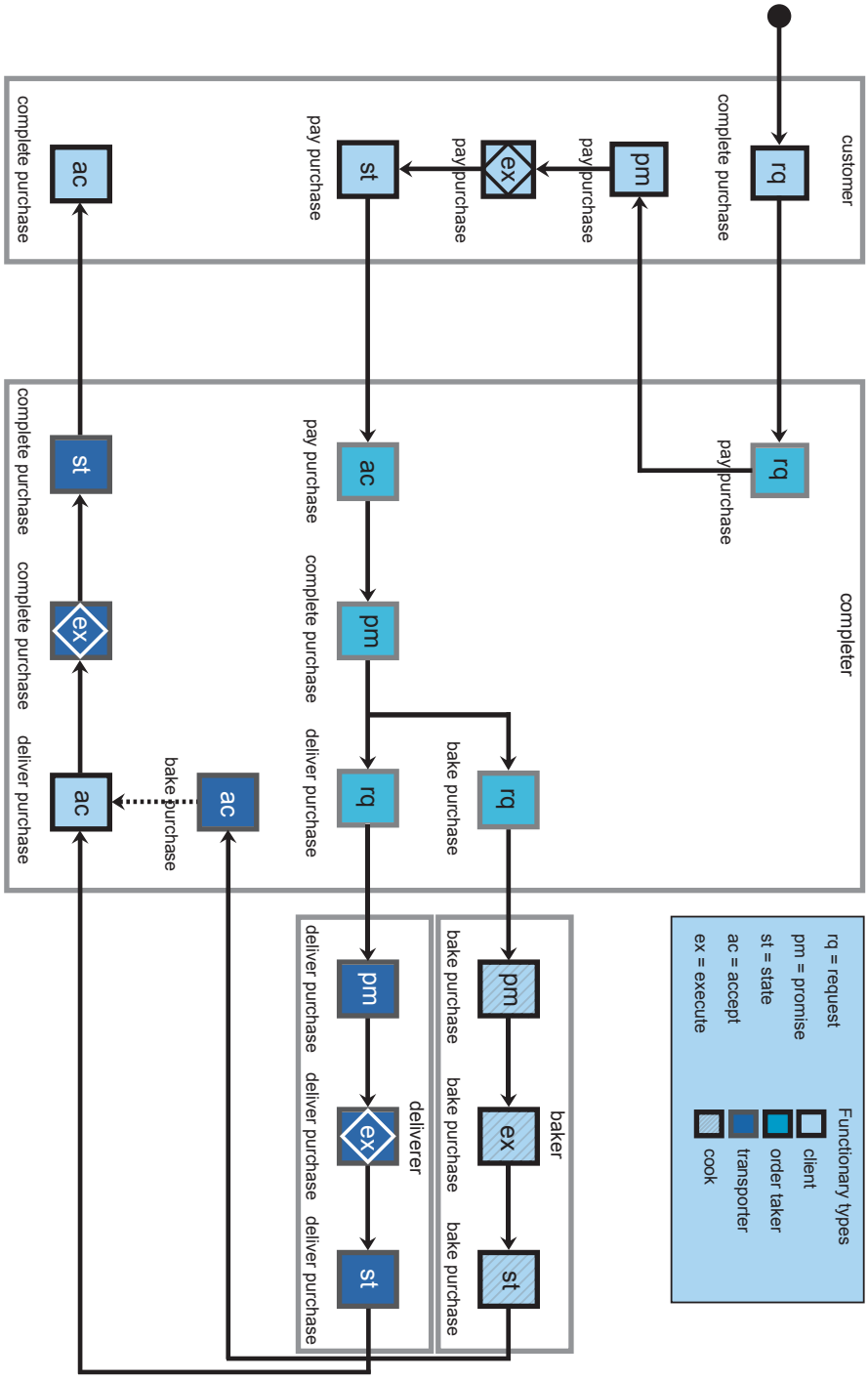

Fig. 4.6 Composed transaction orchestration including functionary types

to the functionary type *transporter*. Delegation does not transfer the responsibility: the order taker remains responsible for the entire transaction.

To support their operations, the pizzeria uses several application services and technical infrastructure services: a website which contains the assortment customers can choose from, electronic map software to plan the optimal route to the customer, accounting software, an external hosting service for the website, a PC for running the accounting software, a GPS navigation device and a credit card terminal. This has been illustrated in Figure 4.7, using the ArchiMate notation.

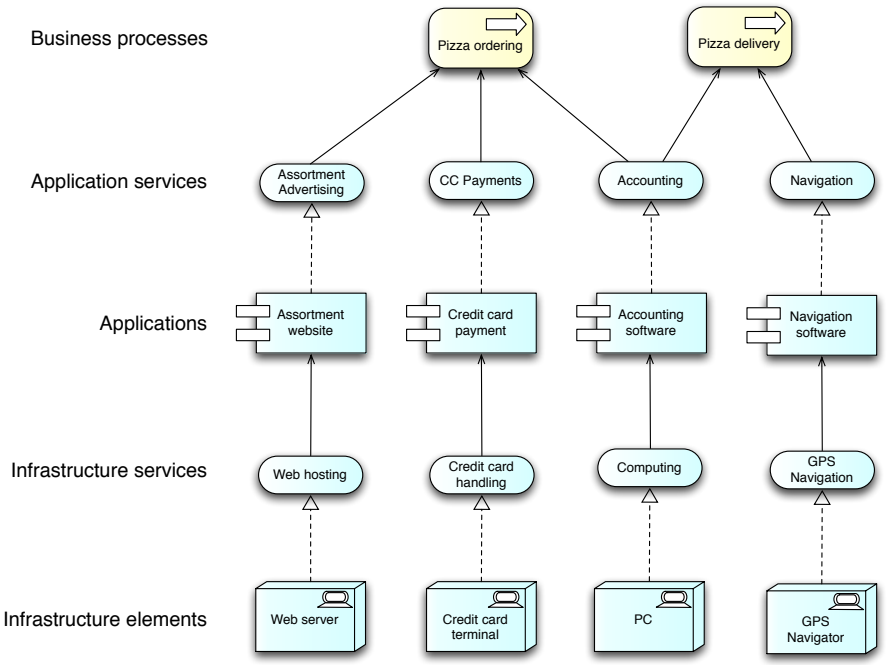

Fig. 4.7 The current software and technical infrastructure services

4.2.2 Intended change

Management of the pizzeria “Perla del Nord” has decided to enter the business-to-business (B2B) market within the next three months, primarily to be able to provide pizza delivery services to offices in and around the city centre. This will enable them to realize the intended growth from 100 pizzas a day to over 150 a day. As an additional service, they will allow B2B customers to pay afterwards by sending them monthly invoices. Before B2B customers are allowed to pay afterwards, their

credit rating is checked first (only B2B customers with a positive rating are allowed to use this service) and a contract must be signed. The pizzeria will use electronic 3rd party services for the credit rating, because they want to dedicate their resources to their core business.

Since the intended change will have its impact on the core processes of the pizzeria, the CEO decides to use an enterprise architecture for the following purposes:

- to speed up the decision making process of the intended change;
- to outline operational and transformational consequences of the intended change in terms of organisational change, human resource policy (e.g. required number of employees, competences), costs and benefits, new business partners. An operational consequence of the intended change is for instance new business transaction for contracting. Transformational consequences are for instance developing a contract template for B2B customers and contracting 3rd party services;
- to determine the best implementation scenario;
- to communicate the intended change to different types of stakeholders: e.g. a view for the CFO which addresses the costs and the necessary additional software or a view for the HR manager that addresses the new competences.

The enterprise architecture, of the desired situation, should be in line with the new strategy. In doing so, the first step is to determine the impact of the principles on the intention of entering the B2B market. Next the impact of the intended change on the construction or the pizzeria organisation is defined.

By entering the B2B market the principle of *Outside own organisation: get money first* needs to be amended to: *Outside own organisation: without a contract, get money first*. New principles are introduced as well: *Focus on core business: baking and delivering pizzas* (business principle), *only control data from own core processes* (information principle) and *prefer using controlled data collection from 3rd parties* (information principle). Since the pizzeria is going to exchange information with 3rd parties, they open their network to the external world. They want to realise this in a secure way leading to a technical infrastructure principle: *controlled access towards 3rd party networks determined at the level of the enterprise service bus (ESB)*.

At the construction level, as illustrated in Figure 4.8, new actor roles are introduced: *B2B-contractant* and *credit rater*. It has to be decided whether this role stays within the organisation of the pizzeria or if this role is to be outsourced or shared. Note that there is no additional actor role introduced for invoicing B2B-customers. We see this as a specific process implementation of the transactions *complete purchase* and *pay purchase*: the request of *pay purchase* for B2B-customers is executed once a month instead of directly after the request *complete purchase*. We also see additional transactions appearing: *rate credibility* and *contract-B2B*. For the credit rating an additional business resource (fact bank) is needed. This fact bank is external, since the pizzeria prefers to use controlled data collection from 3rd parties. We also see that the money/pizza value exchange depicted in Figure 4.1 is now realised in terms of three business services, delivered by three transactions: *complete purchase*, *pay purchase* and *contract B2B*.

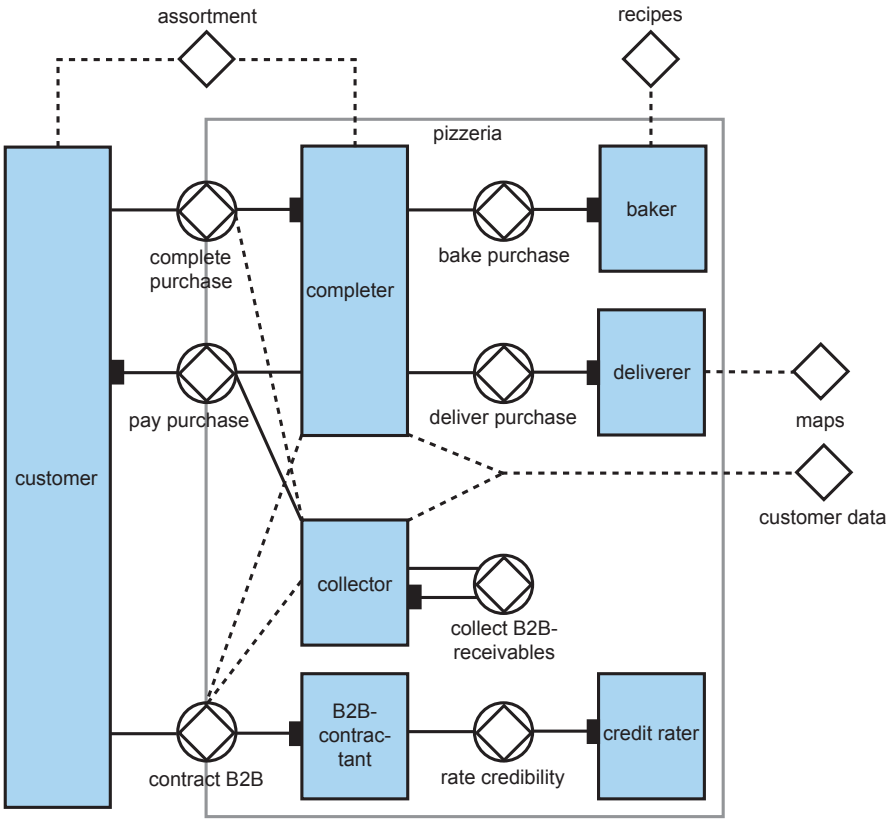

Fig. 4.8 Construction of desired situation

Entering the B2B market yields additional needs to for information and software services: a service to store data of B2B customers (such as contact and delivery information), a service to store the order history of B2B customers as input for e.g. invoices, a service to store the credit rating of B2B customers so the pizzeria does not have to check the credit rating each time the B2B puts an order in, and a service for checking the credit rating of B2B customers at a third party. Finally, a financial service for sending and monitoring the payments of invoices to B2B customers is necessary. The first three services are implemented in a CRM system. The financial service is implemented by using the existing accounting software. For the last service, an external Web service is used.

Entering the B2B market also yields additional needs with regards to technical infrastructure services: a server to run the CRM software (this server will also host the accounting software), an ESB for securely connecting to the credit rating web-service, a local area network to connect the server to the ESB and the PC and a fire-

wall to secure the connection to the Internet. The result is illustrated in Figure 4.9.

To demonstrate the impact of the intended change on the activities of the (functionary type) order taker, the artist impression shown in Figure 4.10 was used in which the changes impacting the work of the order taker have been highlighted. In the new situation the order taker needs to check whether the customer is indeed employed by a company that has a contract with Perla del Nord. Therefore, customers are required to identify themselves using their (company issued) access card and pin number. The order taker must have access to the administration of B2B contracts.

4.2.3 Enterprise architecture results in the Pizzeria case

To finalise the pizzeria case we briefly discuss a non-exhaustive list of specific enterprise architecture results, which can be derived form the pizzeria example. In doing so, we will distinguish different types of deliverables:

- final deliverables;
- intermediary results;
- intangible results.

The first two result types, which are tangible results, are concerned with designing the architecture, communicating the architecture, performing analysis and drafting recommendations. Intangible results are mainly concerned with creating commitment and a shared understanding amongst stakeholders. The pizzeria example yields the following final deliverables:

Models – In this example the Construction Model of the DEMO method has been used. The model presents a univocal, comprehensive, and concrete image of the desired state. The model is an ontological representation of the pizzeria showing the boundaries of the system, the actor roles and the transactions between actor roles. This model can be used to determine the implementation of actor roles in organisations (including sourcing and sharing opportunities), the implementation of the process flow and to identify business services (based on the transactions) including the quality of business services (QoB). E.g. the QoB of the business-service pizza-delivery tells what the bakery will deliver, when the bakery delivers, how much pizzas the bakery delivers and how good the quality of the baked pizzas is. The model can also be used to identify information service, including the quality of information services (QoI). E.g. the QoI of the information service *determine discount* states that the information service must be real-time available, the actuality on the information must be less than 1 minute, meaning that purchases less than a minute ago will not be included in determining the discount.

Stakeholder-specific views – Views are necessary to provide stakeholders with insight in the potential impact on their concerns. E.g. a view for the completer to illustrate the changes in his responsibilities due to the expansion to the B2B

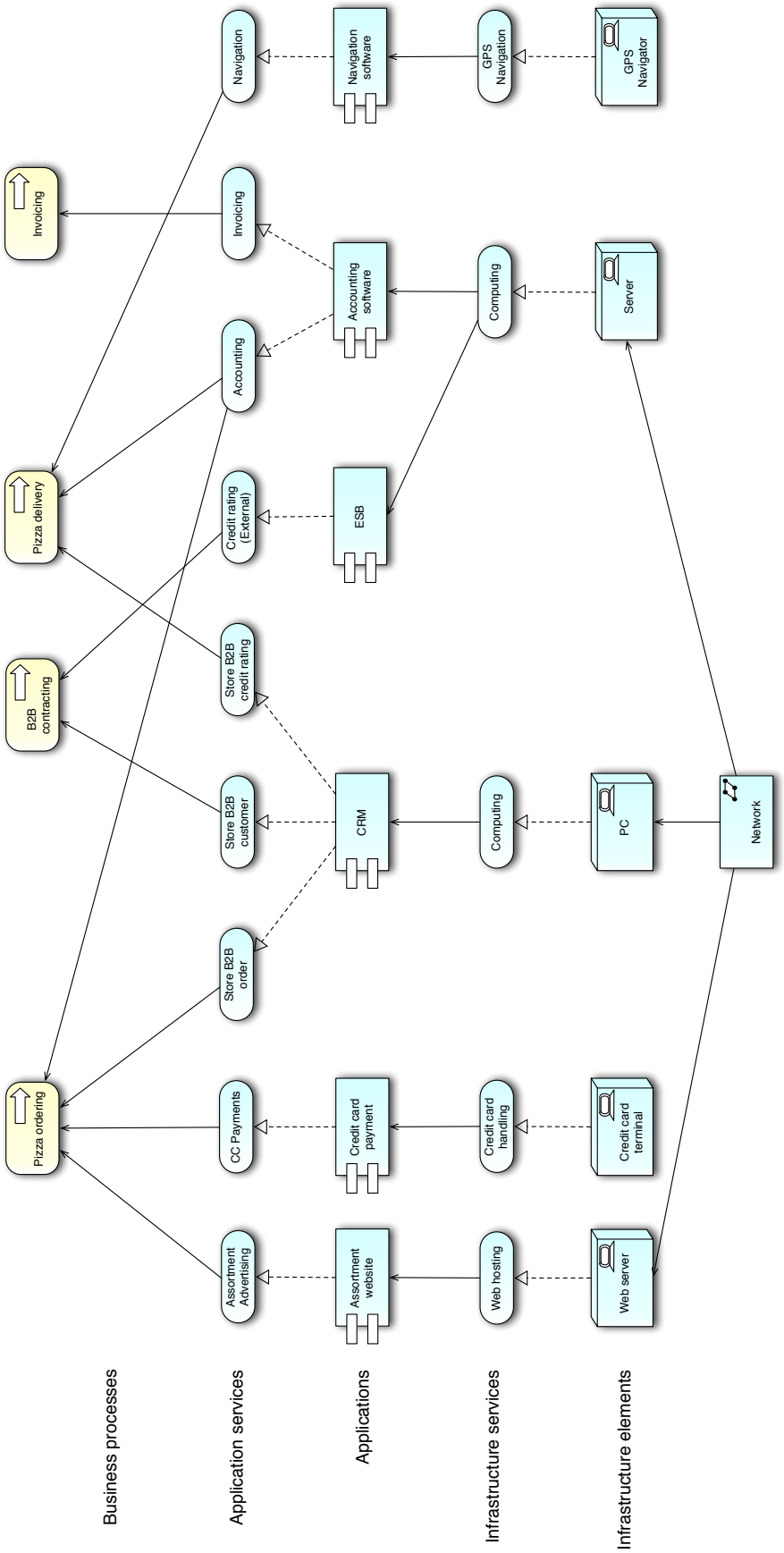

Fig. 4.9 Software and technical infrastructure services of the desired situation

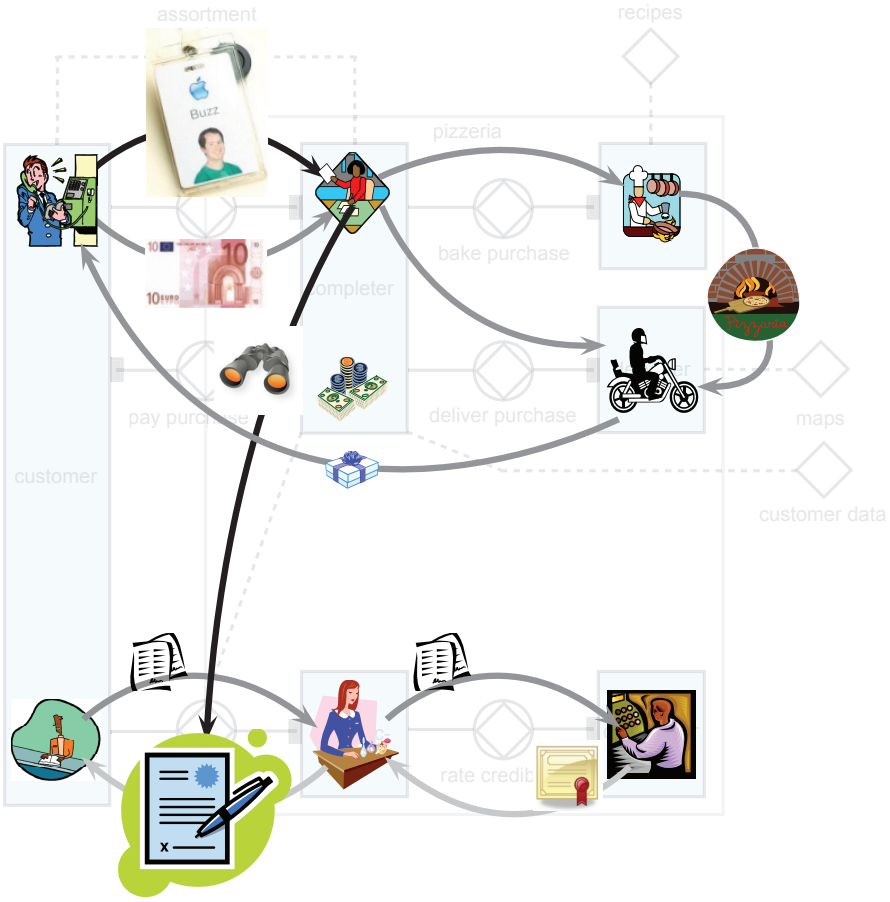

Fig. 4.10 Artist impression of new work environment of order taker

market, a view which shows all internal and external stakeholders or a view showing the connections with 3rd parties which can be used for determining the necessary security measures. Views can take any form, not only narrative descriptions or landscape maps, but also artist impressions, animations, simulations, role plays, games, etcetera.

Specifications and guidelines – The architecture can be used for several specifications such as:

- Determine process flows;
- Define responsibilities;
- Requirement specifications for software based on the QoB and QoI of business- and information services. E.g. the software system used for the

information service *determine discount* should be available real time and during service times with and actuality less than one minute;
- Security guidelines for access towards 3rd party networks.

In the pizzeria example we have seen how the following intermediate results have been produced:

Principles – Principles guided the design of the pizzeria. E.g. the business principle *Outside own organisation: without a contract, get money first* had its impact on the physical implementation of the process flow and the technical infrastructure principle *controlled access towards 3rd party networks determined at the level of the enterprise service bus (ESB)* determined the implementation of security with 3rd party networks.

Solution alternatives – During the design of the pizzeria, some solution alternatives have been identified, for instance a solution alternative where the business services *baking purchase* and *delivering purchase* are being outsourced or a solution alternative where information about the payment history of B2B customers is shared with the *credit-rater* in order to get a discount on their services. The enterprise architecture can be used to determine the validity and feasibility of each solution alternative in order to select the most appropriate alternative.

Traceability of decisions – E.g. the reason why baking and delivering pizzas are executed in parallel can be traced back to the principle of *bake while driving* and the process flow of the pizzeria can be traced back to all business principles.

In the pizzeria example the following intangible results would have played a role (if it were a real case):

Communications amongst involved organisations – The enterprise architecture can be used for communications among organisations involved in the system. E.g. communications between the pizzeria and the 3rd party delivering credibility data as part of contract negotiations: what is the QoI or data ownership;

Commitment amongst stakeholders – Stakeholder specific views, addressing their concerns, will increase their commitment for the intended change.

Even in this simple example of the pizzeria, one can see there are several different results enterprise architecture delivers. The challenge is to determine which products to deliver (when is good, good enough) and to safeguard consistency of interrelations between these products.

4.3 Quality of the produced results

A lot of resources, such as money, time, emotional energy, and intellectual energy are invested into the creation of enterprise architecture results. This raises the question of whether these resources are spent well. *Do the results meet the needs?* In other words: *What is the quality of the results? What is the return on modelling effort (RoME)?*

4.3.1 Possible uses of architecture results

In considering the quality of enterprise architecture products, we take the position that the stakeholders, their concerns, as well as their information, insight and/or steering requirements as a starting point. Some typical examples of such needs are:

Top-level management – Is the intended transformation still justified given the (expected) improvements in relation to the (expected) costs of the transformation? How can we ensure our policies are followed in the development and operation of processes and systems? What is the impact of decisions (on personnel, finance, IT, etc.)?

Middle-level management – The current situation with regard to the computerised support of a business process.

End user – The potential impact of a new system on the activities of a prospective user.

System administrators – The potential impact of a new system on the work of the system administrators that are to maintain the new system.

Operational manager – What new technologies do we need to prepare for? Is there a need to adapt maintenance processes? What is the impact of changes to existing applications? How secure are my systems?

Architect – The requirements of a specific stakeholder with regard to the desired situation. What are the consequences for the maintainability of a system with respect to corrective, preventive and adaptive maintenance?

Project manager – What are the relevant domains and their relations, what is the dependence of business processes on the applications to be built? What is their expected performance?

System developer – What are the modifications with respect to the current situation that need to be performed?In the ArchiMate project [78], three purposes for models and views have been identified:

Designing – supporting architects and designers in the design process from initial sketches to detailed design.

Deciding – supporting decision makers in the process of decision making by offering an insight into the issues/impacts they need to decide upon.

Informing – supporting the informing of any stakeholder about the enterprise architecture and its impact on the future enterprise.

In the ArchiMate project, the focus was on architecture level design models as well as the creation of associated views. Design principles and general business requirements were not taken into consideration. They are, however, also possible results produced in an enterprise architecting effort, albeit not of a design nature. Therefore, it is wiser to generalize the notion of a *designing* purpose to a *specifying* purpose, be it the specification of a design, a set of principles, or general business requirements. Furthermore, developments such as outsourcing stress the purpose of architecture results for contracting reasons. Architecture results have become part of formal contracts governing outsourcing and/or procurement [42]. In sum, we identify four types of goals for which architecture results may be created:

Specifying – making explicit the requirements, principles or designs pertaining to the enterprise, ranging from initial sketches to detailed specifications, including their justification in terms of earlier made decisions.

Deciding – supporting decision makers in the process of decision making by offering an insight into the issues/questions, as well as their consequences and possible justifications, they need to decide upon.

Informing – supporting the informing of any stakeholder about the enterprise architecture and its impact on the future enterprise, possibly including their justification.

Contracting – providing a formal statement of the architectural requirements of enterprise/system components to be realized/worked-out by a supplier.

4.3.2 From intended use to the design of a result

Given this spectrum of uses, the creation of an enterprise architecture results requires deliberate planning. It is not just a matter of coming up with the right content, but also selecting the right depth, subjects, forms, etcetera. In other words, they need careful designing as well. This "result design" involves the identification of the intended use of the result, the intended audience, the subject of the result, and the form to use in representing the result. The subject, and therefore the scope, of the planned result will be dictated by the concerns/interests of the intended audience, and may for example be: value exchanges in the value chain, business services offered, technological infrastructure, detailed process designs, process performance, requirements, principles, etc. The form the result takes refers to the communication style and languages used. One may, for example, opt for an intangible form or a tangible form such as a graphical model/view, a textual model/view, a combination of the latter, or even an animation. For the models one will typically have to make a selection from languages such as:

ArchiMate – a language [78] to express the (design oriented) architecture of enterprises.

DEMO – a language [35, 114] to express the ontology of an enterprise.

e3-Value – a language [46] to express the value exchange between business actors.

UML – even though it [25] is initially designed for software design, has been extended for business modelling [39].

Some of these languages provide pre-defined mechanisms to create views, for example, UML's swimming lanes. The ArchiMate project also defined a number of such mechanisms in terms of landscape maps, process illustrations, etcetera [78]. In addition, one may want to construct ad-hoc views for specific uses.

4.3.3 A deeper understanding of the quality of results

The quality of the results of enterprise architecting efforts is relative to its intended use. Some research has been conducted into the quality of models and the modelling process [24, 74, 75, 76]. Even though not all enterprise architecture results are models, these results can be applied more widely. One of the reasons for this is the fact the notion of *model* is used in a more general way in the research on quality of models than we have defined it in this book. For example, a set of design principles can also be regarded as a *model* representing the intended restriction of design freedom.

Figure 4.11 shows the Sequal framework for model quality as presented in [74]. In this framework a distinction is made between:

Physical quality – How the model is physically represented and available to stakeholders; a matter of *medium*.

Empirical quality – The *comprehensibility* of the model to its intended audience in terms of size, complexity, the number of symbols/graphemes used in a model, etc.

Syntactic quality – Conformity to the syntax of the modelling language.

Semantic quality – How well the model reflects the knowledge (harboured by the domain expert) of the modelled domain.

Social quality – The level of agreement between the stakeholders involved about the model.

Given an intended use of a planned enterprise architecture result, this can be translated to specific requirements with regards to each of the above aspects of model quality. Based on these requirements, an appropriate subject and form for the result can be selected, as well as an effective strategy to create the result [24, 57, 107].

4.4 Enterprise architecture frameworks

As mentioned in the pizzeria Perla del Nord case, apart from establishing which products to produce, it is also a major challenge to safeguard the consistency of interrelations between these products. Architecture frameworks have an important role to play in this. In our view, enterprise architecture frameworks provide:

- a means to order architecture results;
- a means to guard their completeness, both in terms of scoping and level of detail;
- insight into the interrelationships of architecture results, enabling the traceability of decisions and their impact.

In this Section, we therefore provide an exploration of some of these frameworks. The aim of this exploration is to illustrate the range of frameworks. In the next Section we will undertake an attempt to more structurally define the notion of

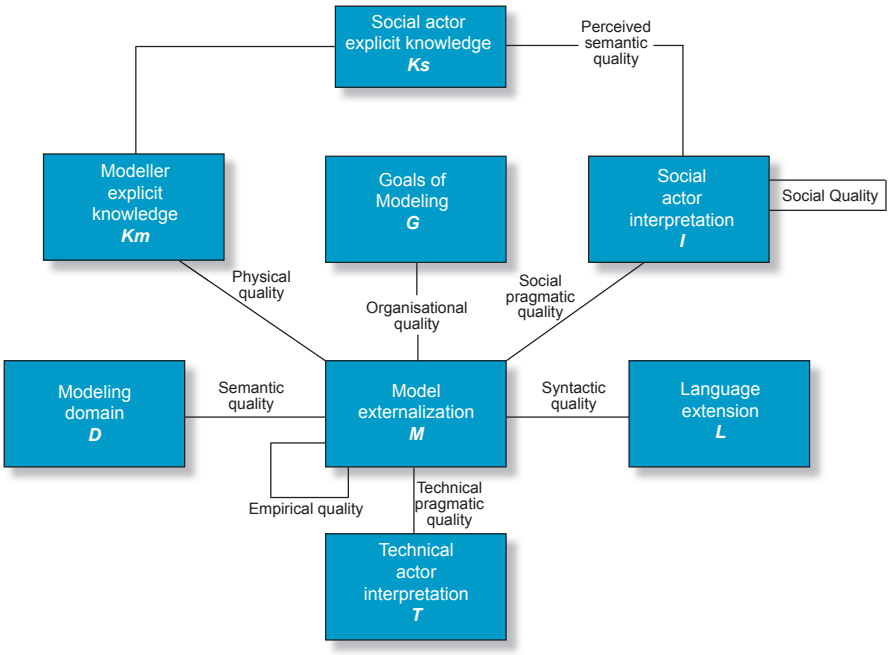

Fig. 4.11 The Sequal framework for model quality

framework in an enterprise architecture context and the possible dimensions these frameworks may occupy.

4.4.1 Tapscott & Caston's views

In [135], the framework as depicted in Figure 4.12 was presented as a way to position different interests with regards to an enterprise and work towards solutions which align these different interests. The focus of each of the views is defined as:

Business view – The *business view* highlights *what* business is conducted by the organisation (or organisational sub-unit), in other words *'the line of business'*. This view considers an organisation as a service providing entity. The business view aims to describe only *what* an organisation does in terms of the services it provides.

Information view – The *information view* provides the information engineering perspective of business solution architectures. The view is concerned with requirements for information resources. This will typically include a definition of what information will be stored, and what business rules this should adhere to.

Work view – The *work view* is concerned with the *how* of the business. This view is expressed in terms of work activities, associated resources, work locations, and needed information. In our context, an important goal for defining a work view is to determine the most effective ways in which the work activities can be supported by IT solutions. An important aspect of the work view is therefore the distinction between manual work and automated work.

Application view – An *application view* describes the business realisation activities that will be automated. It defines how the automated parts of the work view will operate, which information resources are needed, and how technology will be used to achieve this. The application view is positioned in the centre of Figure 4.12 to emphasise the forces that bring about changes to this view. The dominant forces that will change the application view come from the business and the technology sides. Any changes to those views will directly or indirectly result in changes in the application view.

Technology view – The *technology view* focuses on the technology needed to *facilitate* the other components of the architecture. While the business view focuses on *what* an organisation does, this view focuses on *what with*.

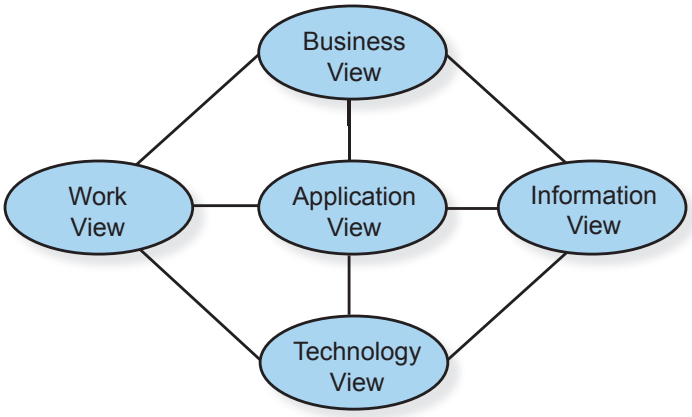

Fig. 4.12 Five views on an enterprise and its IT

Each of the five views from Figure 4.12 defines a specific (basic) subject about the enterprise one is interested in.

4.4.2 The Integrated Architecture Framework.

The Integrated Architecture Framework [30, 45] was developed by Capgemini as a means to structure architecture projects. It has evolved out of several decades of

experience in the field of architecting. The diagram in Figure 4.13 shows the basic structure of Integrated Architecture Framework. The framework is broken down into *aspect areas* (business, information, information systems and technology infrastructure) and *abstraction levels* (contextual, conceptual, logical and physical). To address disciplines of Security and Governance, the framework also recognizes two distinct views dealing with these issues.

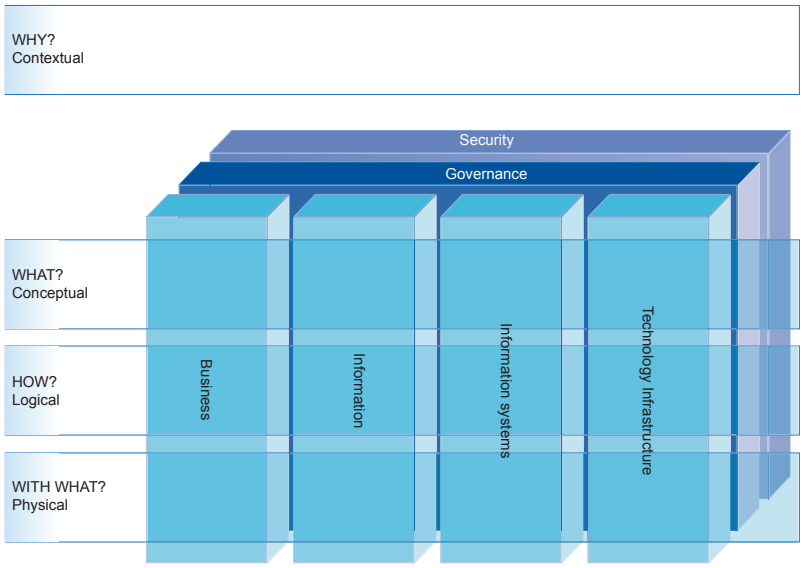

Fig. 4.13 The Integrated Architecture Framework

Abstraction allows a consistent level of definition and understanding to be achieved in each area of the architecture. The Integrated Architecture Framework defines four levels of abstraction:

Contextual level – The contextual level is characterised by "*Why?*" It is not about understanding what the new architecture will be; the level helps to identify boundaries (i.e. scope and objectives) for the new architecture and its context. Specifically, this level focuses on the business aspirations and drivers, capturing the principles on which the architecture will be based.

Conceptual level – The conceptual level is characterised by "*What?*". The requirements and objectives are analyzed and elaborated, ensuring that all aspects of the scope are explored, that relevant issues identified, and resolved, without concern about the way in which the architecture will be realised.

Logical level – The logical level is characterised by "*How?*". The level helps to find an ideal solution that is independent from implementation. From this, several "solution alternatives" can be developed that either provide the same outcome, or alternatively "test" different priorities and scenarios to understand the impli-

cation of different potential outcomes. The outcome of logical level analysis is the vision of the desired to-be state.

Physical level – The physical level is characterised by "*With what?*". It is about determining the real world structure and organisation, and is concerned with translating the logical level 'desired' structure and organisation into an implementation-specific structure, bounded by standards, specifications and guidelines. At the physical level, the outcome is a description of how the desired state will be achieved. The physical level provides standards, guidance and a degree of specifications within which further design will take place.

The Integrated Architecture Framework recognizes four "Aspect Areas", which focus exclusively on the core aspects of the overall architecture:

Business aspect – The business aspect area adds knowledge about business objectives, activities, and organisational structure.

Information aspect – The information aspect area adds knowledge about information that the business uses, its structure and relationships.

Information systems aspect – The information system aspect area adds knowledge about types of information systems (packaged or bespoke) that can automate and support the processing of information used by the business.

Technology infrastructure aspect – The area of technology infrastructure aspect adds knowledge about types and structure of components that support the information systems and actors. These may be hardware or network related. They may include fundamental services such as databases, etc. and key security and other commodity shared services.

The security and governance views respectively focus on:

Governance view – The governance view focuses on manageability and quality of the architecture implementation, required to satisfy the service levels (SLAs) required by the business for its processes and systems. The artefacts for this area are all fundamentally defined within the core aspects areas although the outcome from this aspect area will be new specialized Services and Components to deliver the governance.

Security view – The security view focuses on knowledge to mitigate known risks to the architecture implementation. The artefacts for this aspect area are all fundamentally defined within the core aspects areas. The outcome from this aspect area will be new specialized Services and Components to deliver the required security.

4.4.3 The ArchiMate framework

The ArchiMate project also produced an architecture framework. This framework is depicted in Figure 4.14. The framework identifies a *business*, *application* and *technology* layer, as well as three columns dealing with *passive structure*, *behaviour*

and *active structure.* Even though the distinction between the business, application and technology layers are an integral part of the ArchiMate standard language, the language has been defined in such a way that in principle an arbitrary number of abstraction layers can be stacked which are inter-linked with services, where each layer has the same core concepts as shown in Figure 4.15.

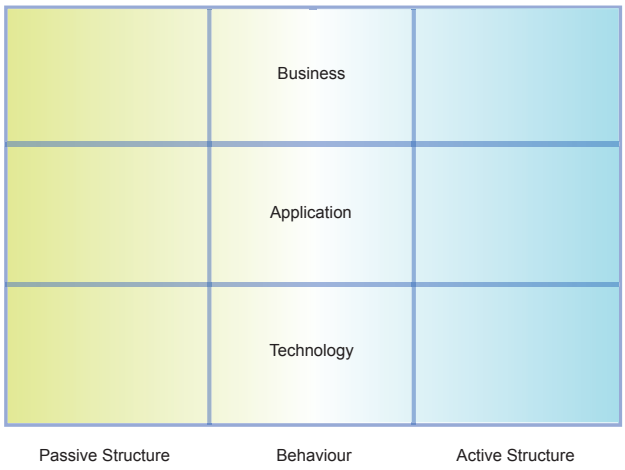

Fig. 4.14 The ArchiMate framework

Figure 4.15 also illustrates the elegance of the ArchiMate language. Rather than offering a plethora of distinct modelling constructs, the language comprises five core modelling concepts (object, service, behaviour element, interface and structure element), which re-appear at each of the layers. In the complete modelling language one will indeed see these five concepts repeated at the three identified levels. However, in essence, they remain the same core concepts. This can be likened to UML's Superstructure [92], where more specific modelling concepts are specialisations from a generic set of basic concepts.

4.4.4 The Zachman framework

The English language, as well as most other languages, contains a class of words called the interrogatives [137]: *Which*, *when*, *how*, *what*, *why*, *where*, *whose*, etcetera. These words may be used to formulate questions concerning situations, people, or any other phenomenon we may perceive or conceive. In other words, we may use these interrogatives to identify different relevant aspects of an enterprise. By using questions based on the interrogative words, insight may be gained into different aspects of an enterprise, such as: *Actors*, *timing*, *processes*, *functionality*, *rationale*, *purpose*, *locality*, *structure*, *ownership*, etcetera. These aspects form the core of the

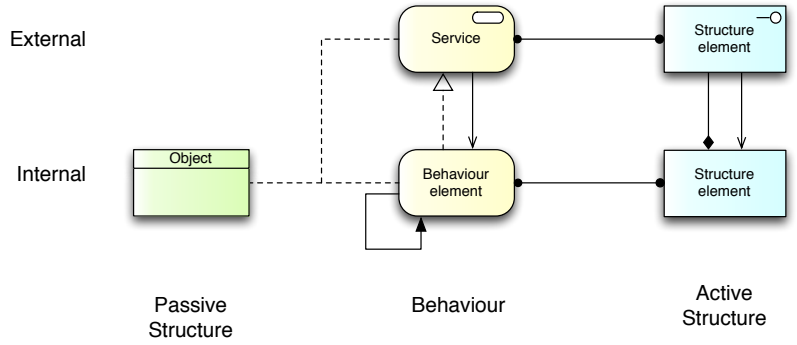

Fig. 4.15 Core concepts of the ArchiMate language

Zachman enterprise architecture framework [155] as depicted in Figure 4.16. The columns cover the following aspects (compiled/taken from [128, 155]):

Data (*what*) – This column addresses data needed for the enterprise to operates, the structure of the data, the way it will be stored, etc.

Function (*how*) – This column is concerned with the operation of the enterprise. It translates the mission of the enterprise into successively more detailed definitions of its operations.

Network (*where*) – This column is concerned with the geographical distribution of the enterprise's activities.

People (*who*) – In this column, one is interested in the people who do the work, the allocation of work, and people-to-people relationships within the enterprise.

Time (*when*) – Time is abstracted out of the real world to design the event-to-event relationships that establish the performance criteria and quantitative levels for enterprise resources.

Motivation (*why*) – The why column is comprises the descriptive representations that depict the motivation of the enterprise, and will typically focus on end-means-end, where ends are objectives (or goals) and means are strategies (or methods).

In addition to the six columns, the Zachman framework identifies six layers (compiled/taken from [128, 155]):

Scope – The first architectural sketch is a "bubble chart", which depicts in gross terms the size, shape, spatial relationships, and basic purpose of the final structure. It corresponds to an executive summary for a planner or investor who wants an estimate of the scope of the system, what it would cost, and how it would perform.

Enterprise or business model – Next are the architect's drawings that depict the final building from the perspective of the owner, who will have to live with it in the daily routines of business. They correspond to the enterprise (business) model, which constitutes the design of the business and shows the business entities and processes and how they interact.

System model – The architect's plans are the translation of the drawings into detailed specifications from the designer's perspective. They correspond to the system model designed by a systems analyst who must determine the data elements and functions that represent business entities and processes.

Technology model – The contractor must redraw the architect's plans to represent the builder's perspective, which must consider the constraints of tools, technology, and materials. The builder's plans correspond to the technology model, which must adapt the information system model to the details of the programming languages, I/O devices, or other technology.

Detailed representations – These correspond to the detailed specifications that are given to programmers who code individual modules without being concerned with the overall context or structure, and/or process designers who design detailed workflows.

Functioning enterprise – Finally, a system is implemented and made part of an organisation. This is a view of the program listings, database specifications, networks, and so forth that constitute a particular system. These are all expressed in terms of particular languages.

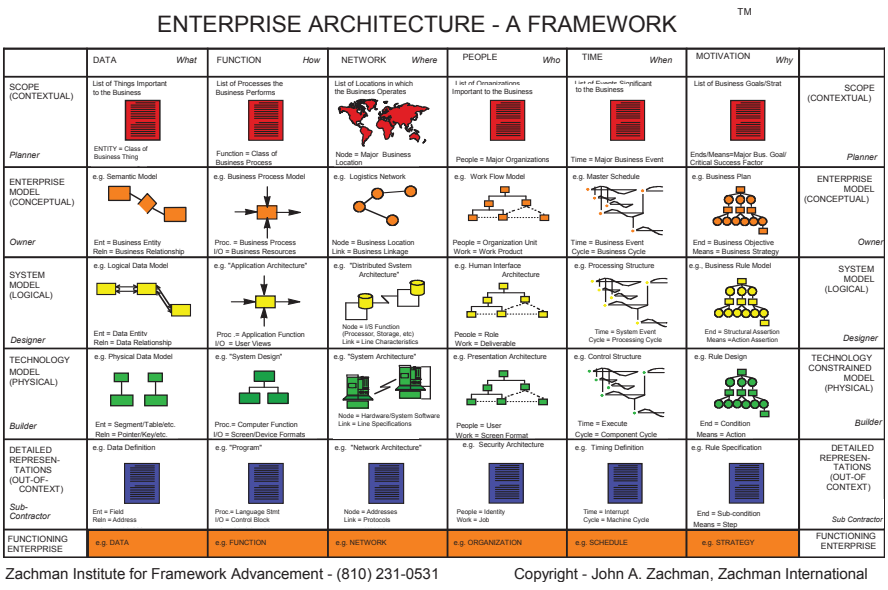

Fig. 4.16 The Zachman Framework

4.4.5 The Open Group's Architecture Framework

TOGAF [139], The Open Group's Architecture Framework, is organised into three Sections: *Architecture Development Method*, *Enterprise Continuum* and *Resource Based*. This is illustrated in Figure 4.17. Each of the Sections provides some guidance on what the outputs of a TOGAF-derived architecture should be and how they should be structured. Below we provide a discussion of these Sections. This discussion is taken from [136].

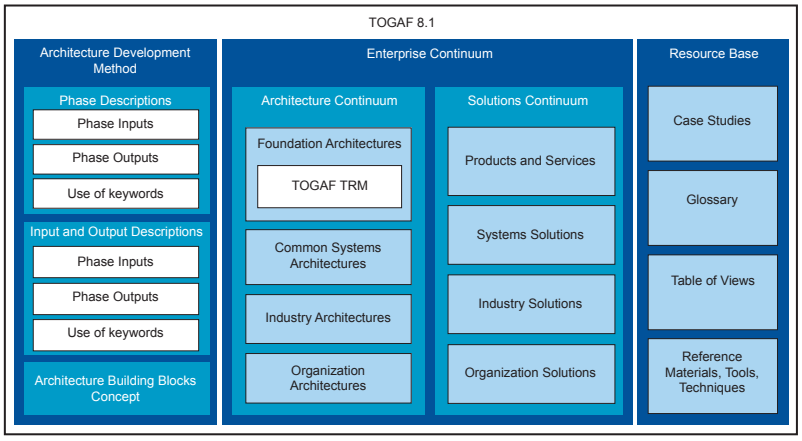

Fig. 4.17 TOGAF 8.1 content overview

4.4.5.1 The TOGAF Architecture Development Method

The *Architecture Development Method* (ADM) explains how to derive an organisation specific enterprise architecture that addresses business requirements. ADM is a major component of TOGAF and provides guidance for architects on a number of levels:

- It provides a number of architecture development phases (e.g., Business Architecture, Information Systems Architectures, Technology Architecture) in a cycle, as an overall process template for architecture development activity;
- It provides a narrative of each architecture phase, describing the phase in terms of objectives, approach, inputs, steps, and outputs. The inputs and outputs Sections provide an informal definition of the architecture content structure and deliverables;
- It provides cross-phase summaries covering requirements management, phase input, and phase output descriptions.

The ADM is primarily a process framework. As such, it can be used in conjunction with different product frameworks, such as the IAF, ArchiMate and Zachman frameworks.

4.4.5.2 The Enterprise Continuum

The *Enterprise Continuum* provides a model for structuring an architecture repository – a "virtual repository" of all the architecture assets. This is based on architectures and solutions (models, patterns, architecture descriptions, etc.) that exist both within the enterprise and in the IT industry at large, and which the enterprise has collected for use in the development of architectures. Architecture Building Blocks reside within the Enterprise Continuum. At relevant places throughout the TOGAF ADM, there are reminders to consider which architecture assets the architect should use.

4.4.5.3 The TOGAF Resource Base

The TOGAF *Resource Base*, "the reference content", is a set of resources, guidelines, templates, background information, etc. provided to be of assistance to the architect in the use of the ADM.

4.5 Dimensions for architecture frameworks

In the previous Section we discussed four architecture frameworks. There are, however, many more frameworks in existence. In addition to Tapscott and Caston, Zachman, IAF, TOGAF, and ArchiMate, some other publicly available frameworks are: the CRIS framework [91], Multiview [153], Kruchten's $4+1$ framework [77], RM-ODP [64], eTOM's business process framework [138], GERAM [61], IAF [30, 45], EEF [98] and DYA [148]. In addition, several organisations use their own internally created architecture framework. Usually, these frameworks have been constructed by their respective authors in an attempt to cover all relevant aspects of the design/architecture of some class of systems/enterprises.

Each of these frameworks covers several dimensions and comprises a number of cells. In the remainder of this Section we will explore several of these dimensions. The list of dimensions is based on work reported in several other publications [42, 49, 56, 64, 67, 77, 78, 91, 101, 104, 105, 107, 135, 139, 138, 148, 153, 154].

Most notably, the authors of [49, 56, 67, 154] have endeavoured to distinguish the dimensions used to span architecture frameworks. Below we provide a synthesis of these latter two works combined with our own observations from the other frameworks in existence. In doing so, we will distinguish three classes of dimensions:

1. Dimensions pertaining to the *subject* of a view.

2. Dimensions dealing with the *purpose* of views.
3. Dimensions concerned with the *form* of views.

When comparing these dimensions in architecture frameworks with the well-known notion of *scope* as used in project/program management, we will see that this notion of scope is potentially related to (selections made in) each of these (classes of) dimensions. Hence the lists provided below can be used to scope the assignment of projects/programs.

4.5.1 Subject dimensions

We now first turn our attention to those dimensions for enterprise (engineering or architecting) frameworks concerned with the subjects of the cells in the framework. Our aim is not to provide an exhaustive list of dimensions, nor do we claim that the list is orthogonal. We merely aim to provide an overview of some of the dimensions used.

Range – The range of the domain that is under consideration. Example classifiers are: single business processes, business units, the entire enterprise, and the entire value chain or ecology.

Construction abstraction – The level of abstraction from the actual construction of the enterprise. As used in the pizzeria example, example classifiers would be: valuation (*what value does it provide to its environment/ecology?*), function (*which functions does it provide in creating this value?*), construction (*how does it realize these functions*?).

Implementation abstraction – The level of abstraction from underlying technologies (including IT, human technology, machines, etc). Example classifiers (inspired by [63]) would be: conceptual (*what is needed?*), logical (*how will it be constructed/composed?*) and physical (*with which technologies and infrastructural elements will it be implemented?*). Note: the construction may use functions provided by other parties within the environment/ecology.

Enterprise system types – When considering an enterprise, several system types can be discerned covering different facets of the enterprise. Some example system-types are: business system, information system, production system, IT infrastructure and management & control system. Most architecture frameworks, in line with their IT roots, identify a dimension based on system types, which focuses on *business realisation through IT*. In these latter frameworks we usually find classifiers (system types) such as: business, information systems, applications and infrastructure.

Aspects of dynamic systems – Enterprises are dynamic systems. In such systems there will be actors/agents which exhibit behaviour, which will impact on objects in the domain. Typical classifiers are therefore: behaviour (*what happens?*), passive structure (*what is it happening to?*), and active structure (*what is doing it?*).

System qualities – Models may focus on different *qualities* of systems. Several quality attributes are in existence [65, 66]. Some examples are: efficiency, security, functionality and reliability.

Interrogatives – The English language, as well as most other languages, contains a class of words called the *interrogatives* [137]. These lead to a natural set of classifiers: which, when, how, what, why, where, whose, ...

4.5.2 Purpose dimensions

With the purpose of an enterprise architecture results, we refer to the combination of the communicative goal of the result, the intended audience and the process context:

Goal of the result – In the introduction of this Chapter already identified four types of goals for which architecture results may be created: specifying, deciding, informing and contracting.

Audience – The audience of the result. Some examples are: decision makers, actors in the transformation (architects, designers and engineers), actors in the current/future enterprise.

Transformation stage – The focus with regard to the stages in an enterprise's development. Some possible classifiers might be: existing situation (as-is), requirements on the future enterprise (as-desired), and its design (to-be).

Planning horizon – The planning horizon with which we regard the (future) enterprise. Typical examples are: current, near-term and long-term.

4.5.3 Form dimensions

As discussed before, the form of an enterprise architecture result refers to the communication style and languages used. One may, for example, opt for an intangible form or a tangible form such as a graphical model/view, a textual model/view, a combination of the latter, or even an animation. The actual form of a result should be in line with its intended purpose and audience.

In the case of intangible results, three dimensions (related to *pragmatic quality* of models [24]) can be applied:

Level of understanding – The level of shared understanding concerning the architecture results.

Level of agreement – The level of shared agreement with regards to the architecture results.

Level of commitment – The level of commitment concerning the architecture results.

The distinction between a level of agreement and a level of commitment may sound artificial. Nevertheless, in practice there is a distinction to be made between

agreeing that something is right for the enterprise, and indeed accepting the consequences these choices will/may have on a stakeholder's goals and concerns. This becomes even more critical when such consequences materialize after the decisions have been made, since the transformation process of an enterprise is likely to impede on the concerns of many stakeholders. An important aspect in the process of enterprise architecting is therefore the creation of a shared conceptualisation of the direction, which the enterprise transformation process should take. This makes it important to obtain the right levels of understanding, agreement and commitment at the right time. The enterprise architecture ideally is the embodiment of this shared conceptualisation.

With regards to tangible results, we may distinguish the following dimensions:

Level of detail – The level of detail to be covered in a result, in other words how detailed the results mimic the intended/existing enterprise. In [78] three levels are suggested: detailed, coherence and overview. Views at a detailed level typically focus on one cell of an architecture framework, while views at a coherence level will generally focus on the relationships between the cells within one dimension. Views at the overview level will aim to provide an overview covering multiple dimensions at the same time.

Level of precision – The precision at which the results are specified. A possible way to express the level of precision would be in terms of its level of formality, referring to the level at which it would allow for mathematical/automated interpretation and/or manipulation. Some example levels would be [101]: informal, semi-formal and formal. Informal would typically be a graphical sketch or a loose narrative description. Semi-formal would be using a controlled (graphical or textual) language, i.e. limiting the allowed syntactic variation, yet still without a well-defined semantics. Formal then implies the use of a (restricted) language with a well-defined semantics, enabling a precise and unambiguous interpretation of the results.

In colloquial use, the levels of detail and precision tend to be confused. When representing something at an overview level, one tends to misinterpret this as an excuse to provide a vague and imprecise description.

4.6 A methodical perspective on the creation of results

Enterprise architecture results will typically feature numerous models and views. In an architecture context, these models are created in line with so-called viewpoints [60]:*A specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis.* To some extent a viewpoint establishes a *method* for the creation of views, in particular where it concerns "*the techniques for its creation and analysis*". In [122, 152] a framework was proposed to dissect a modelling method into a number of aspects:

Way of thinking – Articulates the assumptions on the kinds of problem domains, solutions and modellers. This notion is also referred to as *die Weltanschauung* [126, 153], *underlying perspective* [83] or *philosophy* [12].

Way of modelling – The way of modelling provides an abstract description of the underlying modelling concepts together with their interrelationships and properties. It structures the models, which can be used in the information system development, i.e. it provides a language in which to express the models.

Way of working – Structures (parts of) the way in which a system is developed. It defines the possible tasks, including sub-tasks, and ordering of tasks, to be performed as part of the development process. It furthermore provides guidelines and suggestions (heuristics) on how these tasks should be performed.

Way of managing – The managerial aspects of system development. Originally this was referred to as the *way of controlling*. It includes such aspects as human resource management, quality and progress control, and evaluation of plans, i.e. overall project management and governance (see [72, 127]).

Way of supporting – The support to system development that is offered by (possibly automated) tools. In general, a way of supporting is supplied in the form of some computerized tool.

The resulting framework is shown in Figure 4.18. As synonyms, one may refer to a way of working as a *(modelling) approach* and to a way of modelling as a *(modelling) technique*. The *way of thinking* and *way of modelling* of a method are strongly tied to the subject dimensions in architecture frameworks since these dimensions determine *what* is to be modelled. The *way of modelling* is also tied to the form dimensions, in particular the levels of detail, specificity and formality needed in the results.

When using an architecture framework, a large number of viewpoints can be associated. Starting with the subject dimensions:

- For each cell in the framework there is likely to be a specific viewpoint. For example: the business aspect at the conceptual level, or the technology aspect at the logical level.
- For some complete dimensions, there will be viewpoints covering the entire dimension. For example, the entire conceptual level.
- There are likely to be viewpoints focusing on the relationships between a cell or a dimension, and its direct neighbours. For example, relating the conceptual level to the logical level, or the business aspects to the informational aspects.

In addition, mainly fuelled by the purpose, there are likely to be many, many, ad-hoc viewpoints depending on specific concerns, audiences and purposes. For example, a view showing a design alternative at the physical level, with an associated cost/benefit analysis at the conceptual level, aiming at answering the concern of the CFO.

In creating/selecting an architecture framework, organisations should focus on a framework that is based on structural considerations based on the stakeholder's concerns that are key to the enterprise and require enduring attention. Ad-hoc viewpoints (and ensuing views) can always be added when needed. This most likely

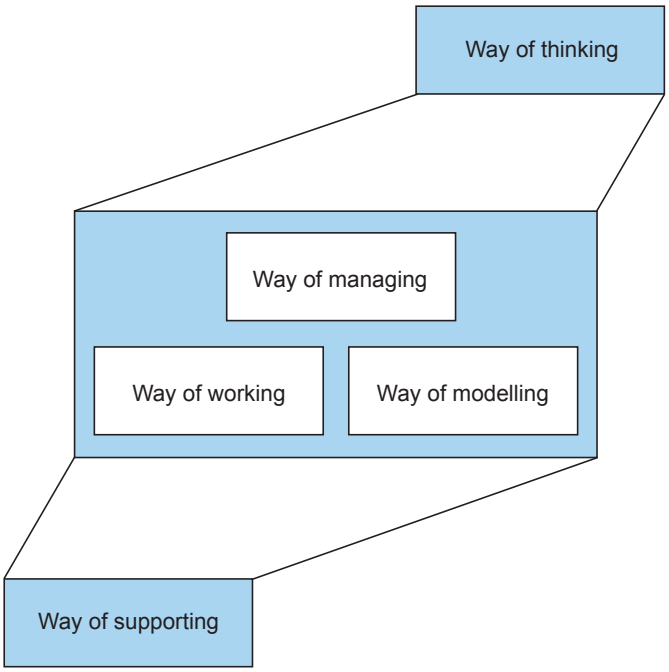

Fig. 4.18 Aspects of a modelling method

implies that the framework will primarily comprise subject dimensions, focusing on view(point)s providing stakeholders with insight, as well as steering abilities, relevant to their concerns. Additional ad-hoc view(point)s can be added on top of that tuning the structural view(point)s to the specific needs/uses at hand.

4.7 The call for a unified notation

As mentioned in the pizzeria example, there is no single modelling language in which to model all relevant aspects of the pizzeria domain with a unified look-and-feel. In the pizzeria example, we showed several aspects of this organisation using modelling techniques such as: e3Value, DEMO and ArchiMate. Even though it is doubtful whether it is possible to have a single unified language covering all aspects of an enterprise, a well integrated language with a unified look and feel covering at least the core valuation, function and construction aspects of an enterprise is desirable [78]. Note that in expressing this desire, we refer to the *modelling* language used to create *models* of different aspects of an enterprise. When creating *views,* one is likely to use different styles and languages that are better attuned to the needs of the intended audiences. The models will be used primarily by architects and engineers to express the design of the enterprise in all of its richness.

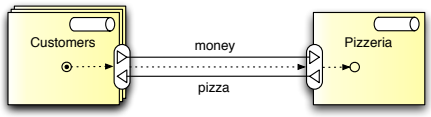

Fig. 4.19 Pizzeria at the valuation level

ArchiMate was designed to play the role of such a unified language. Even though we only use the ArchiMate notation for the infrastructural aspects of the pizzeria, it is indeed suitable to represent the construction diagrams used in the pizzeria case. Even more, the ArchiMate language allows for the definition of extensions, where new concepts are introduced as specialisation of existing ones and/or combinations of existing ones. Even though the concept of transaction, as used in our Pizzeria example, is not explicitly built-in the ArchiMate language, it can be constructed by defining a transaction concept being a collaboration between the two roles participating in the transaction, involving a number of processes (covering the request, promise, execute, state and accept phases) and associated orchestration. The same applies to the value exchange between the customer and the pizzeria, which can essentially be constructed in terms of ArchiMate's value and service concepts. When using such constructs, one would be able to produce diagrams such as shown in, Figure 4.19, Figure 4.20, Figure 4.21, Figure 4.22 and Figure 4.23, depicting an ArchiMate-ish version of Figure 4.1, Figure 4.2, Figure 4.3, Figure 4.4 and Figure 4.5 respectively. Note that Figure 4.21 and Figure 4.23 also illustrate how a slightly different perspective leads to additional insight. The original views took the *roles* as leading, by showing how the roles participate in transactions, while the flow of the transactions meanders over them. In the new views we took the *transactions* as leading, and show how the transactions are executed by the different roles.

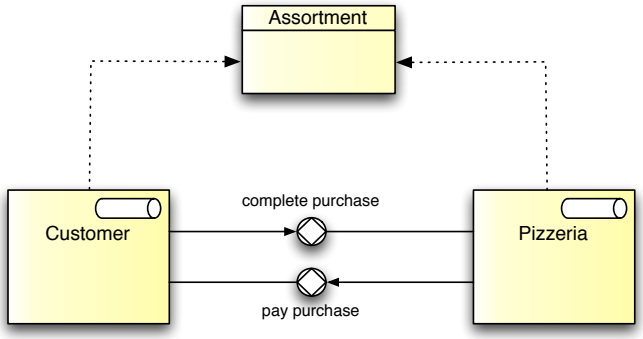

Fig. 4.20 Transactional view of value exchange

Figure 4.19 shows the value exchange between the customer and the pizzeria as a value exchange between business roles. Figure 4.20 shows the realisation of this

value exchange in terms of the transactions *complete purchase* and *pay purchase*. Figure 4.21 depicts the orchestration of the complete purchase and pay purchase transactions as being a collaboration between two business roles: *customer* and *pizzeria*. The construction of the pizzeria in terms of a completer, baker, transporter and their mutual transactions is shown in Figure 4.22. Finally, Figure 4.23 shows the orchestration of the transactions when the actual construction of the pizzeria in terms of the completer, baker and transporter is considered.

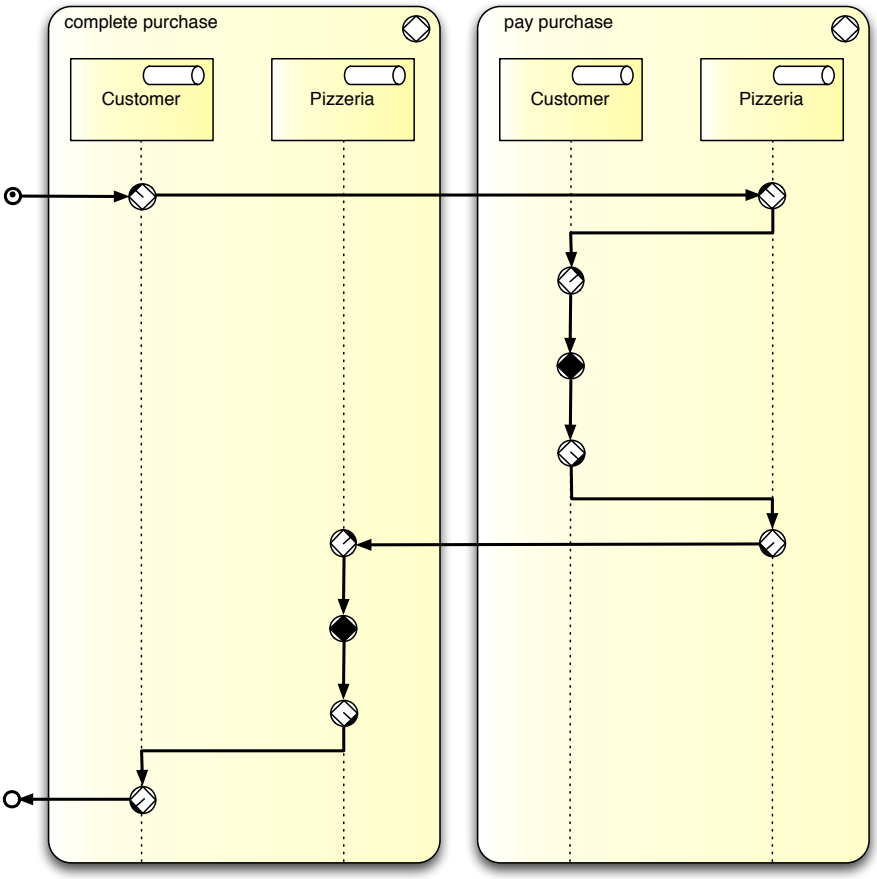

Fig. 4.21 Orchestration of transactions

In the further evolution, and possible integration, of modelling languages such as ArchiMate, DEMO and e3Value, there are some wise lessons to be learned from the development of a language such as YAWL (Yet Another Workflow Language [4]). In the development of this workflow language, the authors started by surveying patterns used in workflow specifications. In doing so, they first gained insight into the

constructions, which the language should be able to express naturally. For example, the desire to discuss/express the impact which a business principle such as "*Outside own organisation: Get money first!*" will have on the orchestration of the transactions between a customer and the pizzeria, requires a modelling language supporting an explicit notion of a transaction and the phases (request, promise, execute, state, accept) within a transaction. The same applies concerning the modelling of value exchanges between the pizzeria and the consumer. The desire to be able to reflect on the value being exchanged between the pizzeria and its consumers, or between any pair of business roles for that matter, requires explicit modelling constructs.

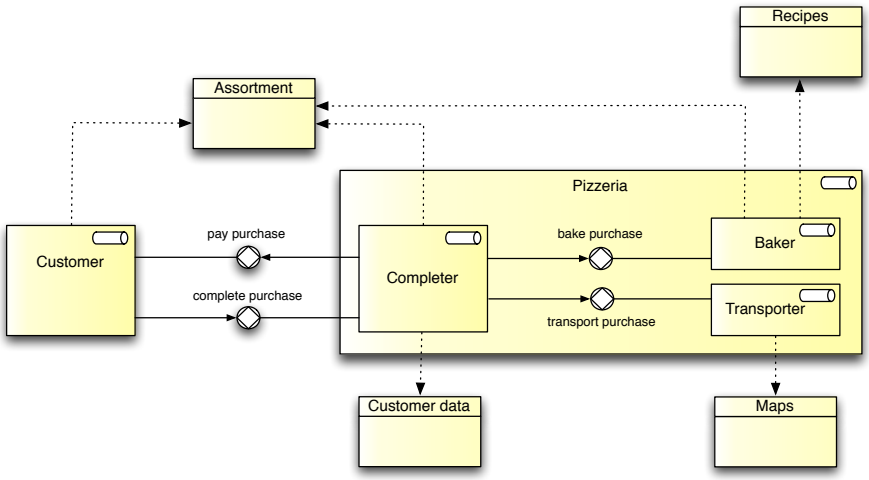

Fig. 4.22 Decomposition of the Pizzeria

When the ArchiMate language was developed initially, it was also based on a survey of needs voiced by the industrial partners in the project [69, 70]. In the meantime, however, insight into what one wants to do with enterprise architectures has evolved, and as a result also the demands on the modelling language used have evolved. In anticipation of such evolution, the ArchiMate language was equipped with an extension mechanism to indeed cater for such evolutions.

4.8 Summary

In this Chapter we have explored the results that may be yielded from enterprise architecting efforts. We have used the pizzeria example to exemplify some of these results, covering final deliverables, intermediary results and intangible results. Tangible deliverables/results include models, principles and views. We then moved on

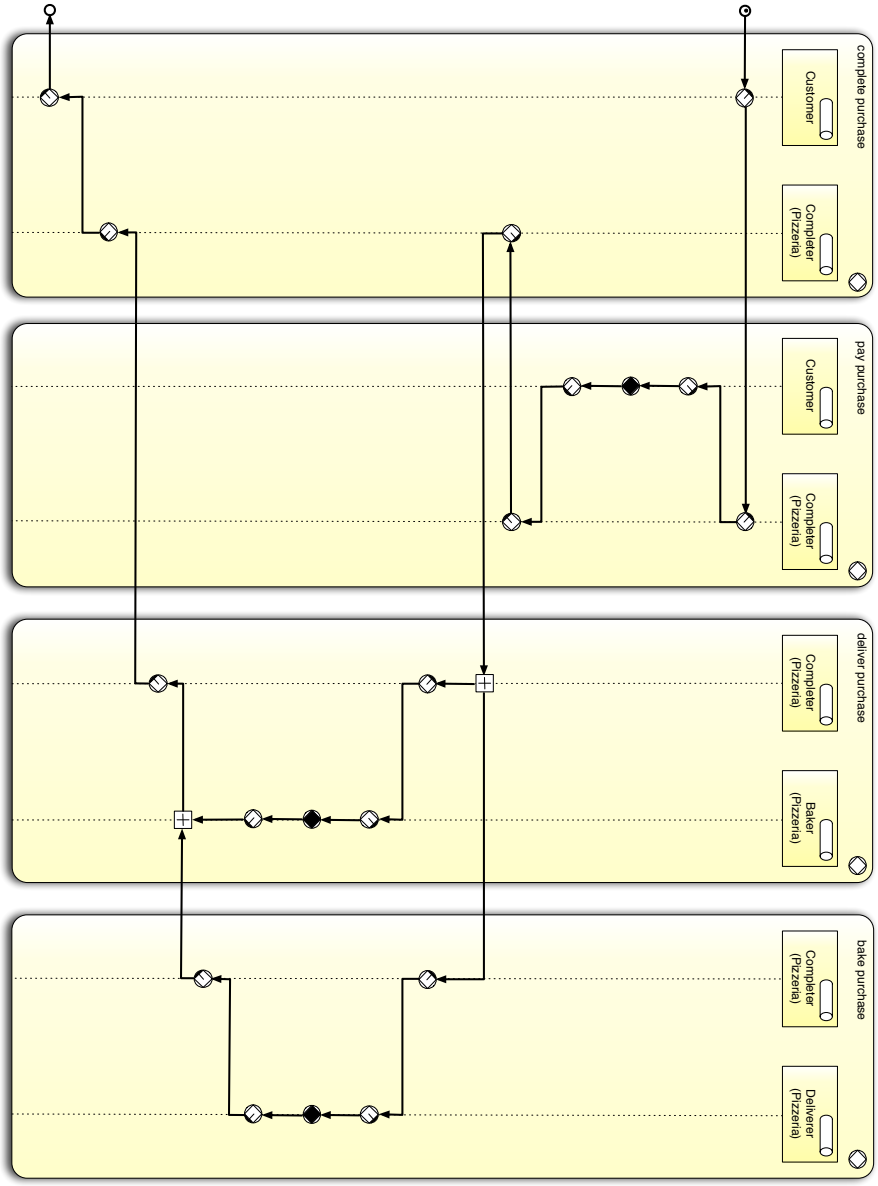

Fig. 4.23 Composed transaction orchestration

to the issue of quality of the possible results. Depending on the intended use, different quality requirements should be met. We identified four key types of usage of results: specifying, deciding, informing, and contracting, also covering justification of decisions made. This intended use has a great impact of the scope of the architecture results needed. Based on the usage requirements of the involved stakeholders and their concerns, the intended result can be designed in terms of its subject and form.

From our desire to have order and completeness in architecture results, we then turned our attention to architecture frameworks, and started by discussing some examples of the architecture frameworks in existence. As means to order architecture results and guard their completeness, a framework gives insight into the interrelationships of architecture results, and therefore enables the traceability of decisions and their impact.

To better understand the concepts underlying the multitude of frameworks, we surveyed some possible dimensions for frameworks, distinguishing between dimensions dealing with the subject, purpose and form. We also related the concept of viewpoint to a pre-existing framework for modelling methods distinguishing between a way of thinking, a way of managing, a way of supporting and a way of modelling. A strong link exists between the subject dimensions of an architecture framework and the way of thinking and way of modelling of the modelling methods used in creating the results, as well as between the form dimensions and the way of modelling of these methods. Even more, we took the position that an enterprise architecture framework should primarily focus on subject dimensions. We also identified an opportunity to define *scoping* as a project/program management notion in terms of the selection of different (classes of) dimensions.

Finally, we argued the case for modelling techniques for (the design-oriented perspective of) enterprise architectures that cover different aspects of an enterprise, while still offering a unified look and feel.

4.9 Discussion statements

1. Views that properly address the concerns of stakeholders are of more value than the underlying architecture as a whole.
2. Artist impressions are more effective than detailed models.
3. An enterprise should standardise on one architecture framework and stick to it.
4. When copying the enterprise architecture of a competitor, one also imports its underlying vision and strategy. In other words, if an enterprise wants to have a unique competitive edge, it needs its own architecture.