

Chapter 12

Case Studies

Hans Bosma, Henk Jonkers, Math J. Cuvelier, Peter G.M. Penders, Saco F. Bekius, and Maria-Eugenia Iacob

12.1 Process and Application Visualisation at ABP

ABP is a pension fund for employees of the Dutch government and the educational sector. ABP is one of the largest pension funds in Europe with total assets of more than 156 billion euros serving more than two million customers.

In the year 1998, triggered by the millennium problem, ABP realised that a better grip was needed on the increasing complexity of the ICT situation. ABP decided to start an information planning and architecture program. Several products came out of this project, as follows:

- Architecture principles, such as:
 - a process starts with a client and ends with a client;
 - every process has a process owner;
 - the organisation works client-oriented;

H. Bosma
Ordina, Nieuwegein, The Netherlands

H. Jonkers (✉)
BiZZdesign, Enschede, The Netherlands
e-mail: h.jonkers@bizzdesign.com

M.J. Cuvelier
Retired, Heerlen, The Netherlands

P.G.M. Penders
Crescimento Universal, Amsterdam, The Netherlands

S.F. Bekius
Dutch Tax and Customs Administration, Apeldoorn, The Netherlands

M.-E. Iacob
University of Twente, Enschede, The Netherlands

- ABP should not ask for certain information from clients if this information is already available within the organisation.
- An architecture vocabulary.
- An information systems blueprint to guide ICT development.

Some of the examples below are in Dutch, because they are taken from real-life data, but they serve to illustrate the type of diagram we are discussing.

12.1.1 ABP Meta-model

ABP divides the architectural universe of discourse into five domains: business, process, application, data and technology (Fig. 12.1). A further detailing in the form of a conceptual meta-model is shown in Fig. 12.2. Notice that the technology domain has not yet been covered by this meta-model.

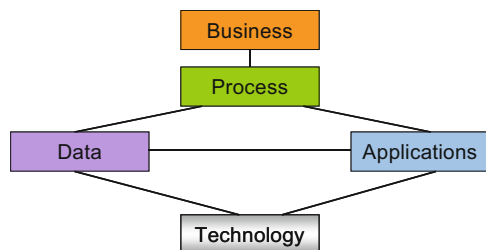
From a first rough comparison with the ArchiMate meta-model (see also Fig. 5.4), the following differences can be identified:

- ABP uses a fixed decomposition of processes and systems, while ArchiMate uses a variable decomposition mechanism.
- The service concept is not used by ABP (although the *process implementation* concept comes close).
- The data domain of ABP has a more extensive set of concepts than the data domain of ArchiMate.
- ABP has no organisational concepts in its meta-model.

12.1.2 Case Essentials

ABP realised that keeping track of the current situation is an important requisite for disciplined ICT management. For that reason, ABP selected a repository in order to store metadata about ICT and in a later phase about other domains. The meta-model of Fig. 12.2 is used as the database scheme of this repository.

Fig. 12.1 ABP architecture domain model



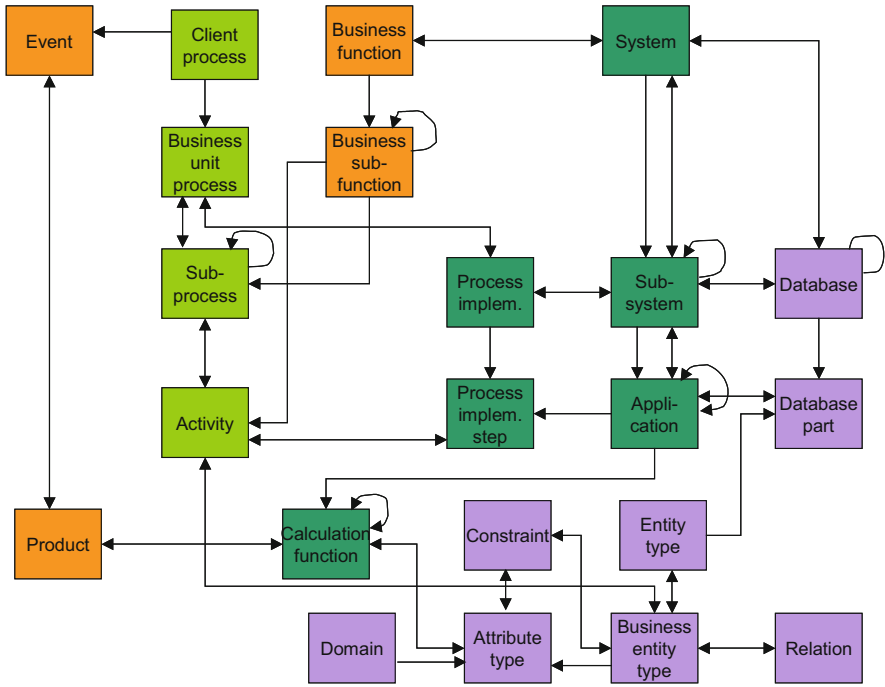


Fig. 12.2 Meta-model of ABP

The repository data is disclosed via a Web portal. However, a graphical presentation of the contents was missing, impeding the wider use of this information. Presentations could be made manually of course, but this requires a considerable effort. Therefore, ABP recognised a need for automatic generation of visualisations. To this end, a tool was built that generates visualisations of the data about information systems, interfaces, and databases. A typical example of such a diagram is shown in Fig. 12.3. For more information about this first ArchiMate case, we refer to Iacob and Leeuwen (2004).

Being able to visualise system information, ABP’s next wish was to connect systems data with process data. However, information about processes was not yet stored in the repository: this process data was stored in a process modelling tool and in a workflow tool. Thus, the goal of the case was to integrate data from different sources and subsequently generate visualisations. The tool infrastructure was to be based on the generic ArchiMate concepts; hence an extra requirement was added, namely to map the (relevant parts of the) ABP meta-model to the ArchiMate meta-model.

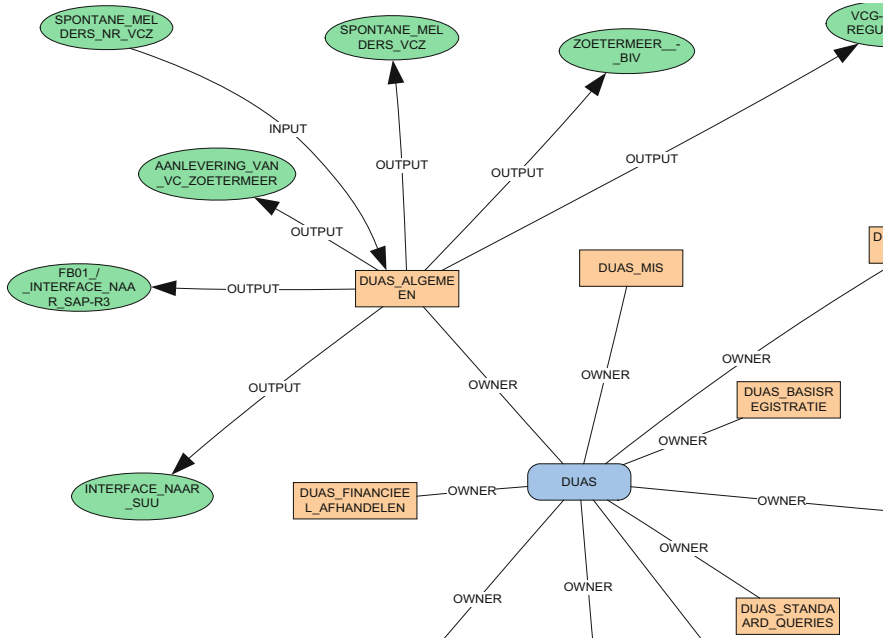


Fig. 12.3 Example of a generated system structure diagram (partial view)

12.1.3 Concepts

To connect system and process information, only part of the ABP meta-model was relevant (Fig. 12.4). The mapping of the ABP meta-model to the ArchiMate meta-model was achieved via an intermediary bridging level, a specialisation of the ArchiMate meta-model. Subtypes of *process* and *application component* are introduced, and the service concept is mapped onto *process implementation*.¹

The concept mapping is depicted in Fig. 12.5. Note that the horizontal *use* relations are derived relations, based on the more detailed usage of *services* of an *application component* by a *business activity*.

12.1.4 Viewpoints

From the four types of design viewpoints identified in Sect. 8.5, the *Composition* and *Support* viewpoints are relevant to visualise the integrated process and system information.

¹Actually, the relation between the concept *process implementation step* and the ArchiMate concept of a *service* is an indirect relation: the existence of a *process implementation step* is only an indicator that an *application service* is provided by an *application component*.

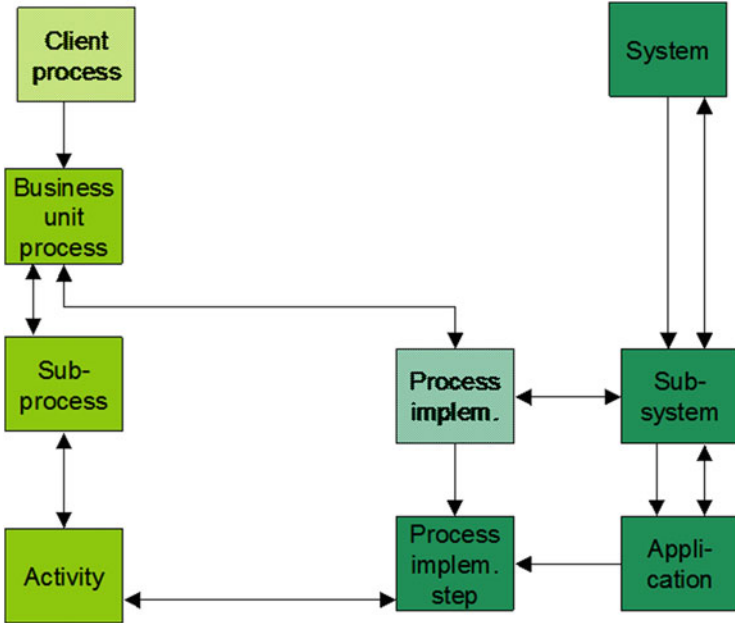


Fig. 12.4 Relevant parts of the ABP meta-model

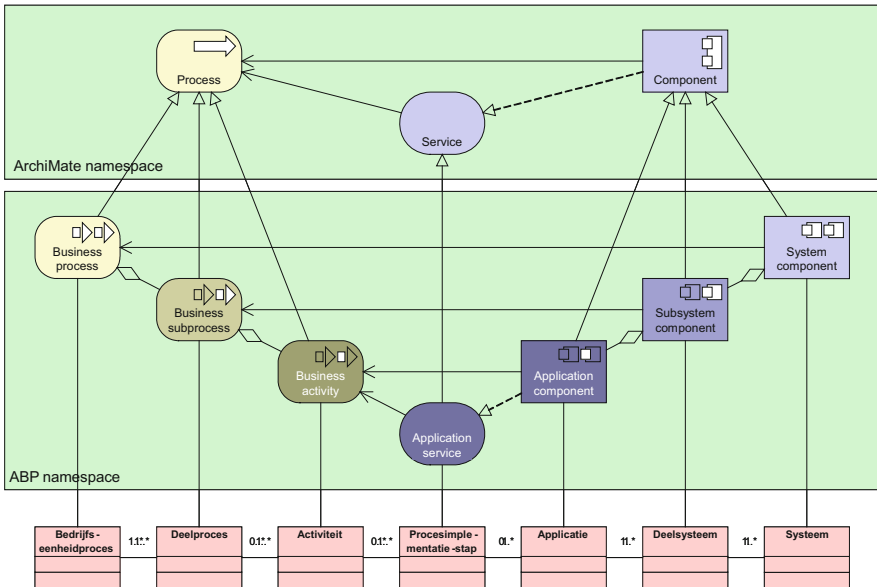


Fig. 12.5 Connecting the ArchiMate meta-model with the ABP meta-model

Composition Viewpoints The Composition viewpoints focus on the structure of processes and systems. The ‘System component composition’ viewpoint shows a *system component* or *subsystem component* and the *subsystem components* or *application components* it consists of (Fig. 12.6). The ‘Business process composition’ viewpoint shows a *business process* or *sub-process* and the *sub-processes* or *activities* it consists of (Fig. 12.7).

Support Viewpoints The Support viewpoints show the usage relations between processes and applications. The ‘Business process dependencies’ viewpoint shows a *business process*, *sub-process* or *activity* and the *system*, *subsystem* or *application components* it uses (Fig. 12.8). The ‘Application component use’ viewpoint shows a *system*, *subsystem* or *application component* and the *business process*, *sub-process* or *activities* it uses (Fig. 12.9). The ‘Process–component relation’ viewpoint zooms in on a particular *use* relation between a *system*, *subsystem* or *application component* and a *business process*, *sub-process* or *activity* (Fig. 12.10).

12.1.5 Design of the Visualiser

The overall design of the visualisation tool for ABP was based on the general workbench architecture described in Sect. 11.4. A logical first step in the integration of the different data sources mentioned in the previous section would have been to add the process data to the repository and integrate it with the systems data. However, integrating data from these different sources proved to be a cumbersome affair because of model and naming incompatibilities. Therefore, the decision was made to concentrate on a subset of the data and use a temporary data store for the

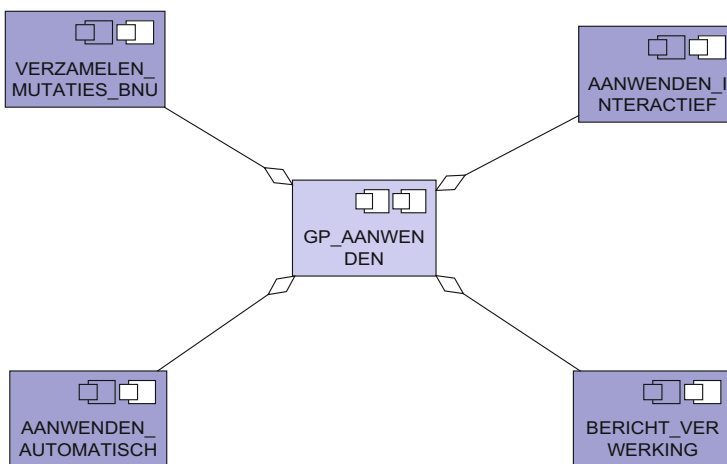


Fig. 12.6 A ‘system component composition’ view

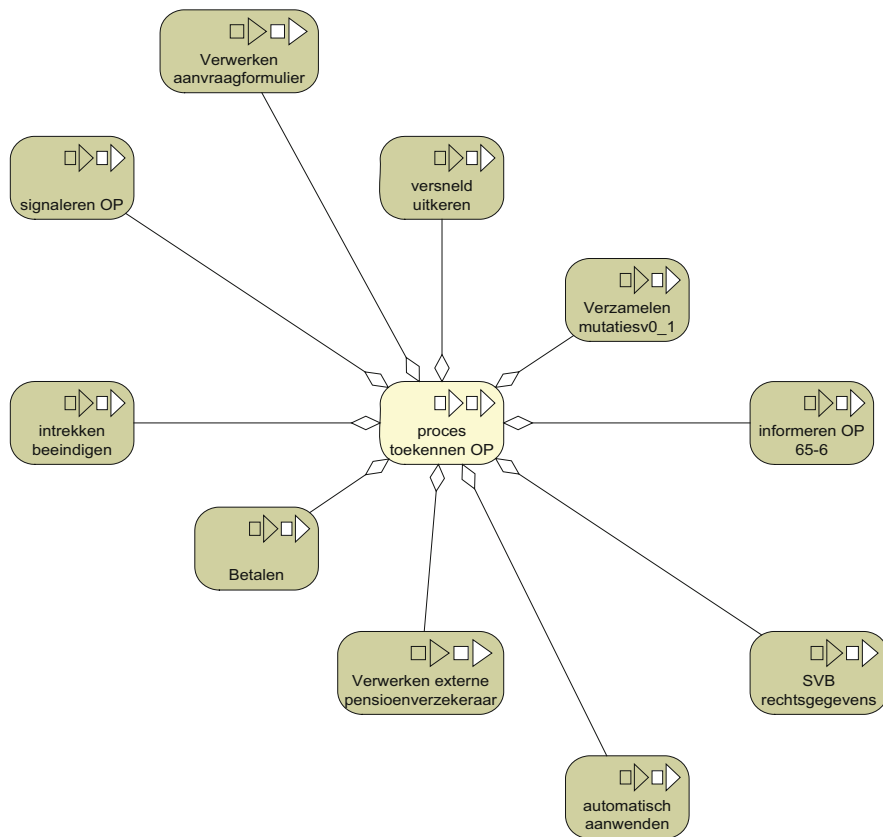


Fig. 12.7 A 'Business process composition' view

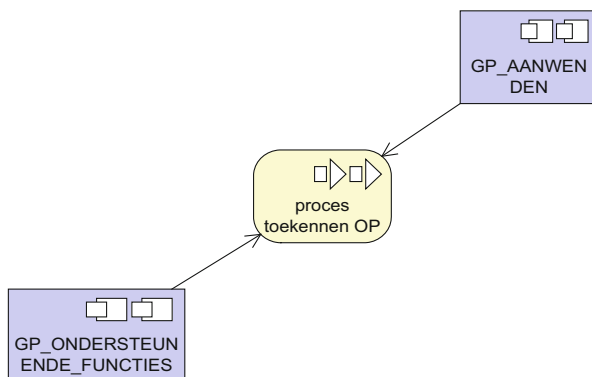


Fig. 12.8 A 'Business process dependencies' view

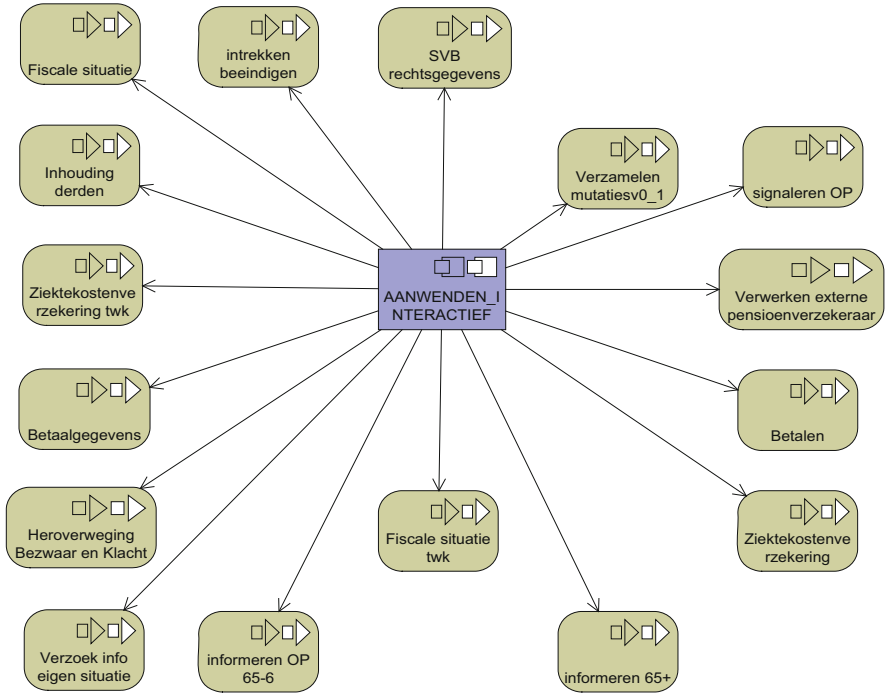


Fig. 12.9 An ‘Application component use’ view

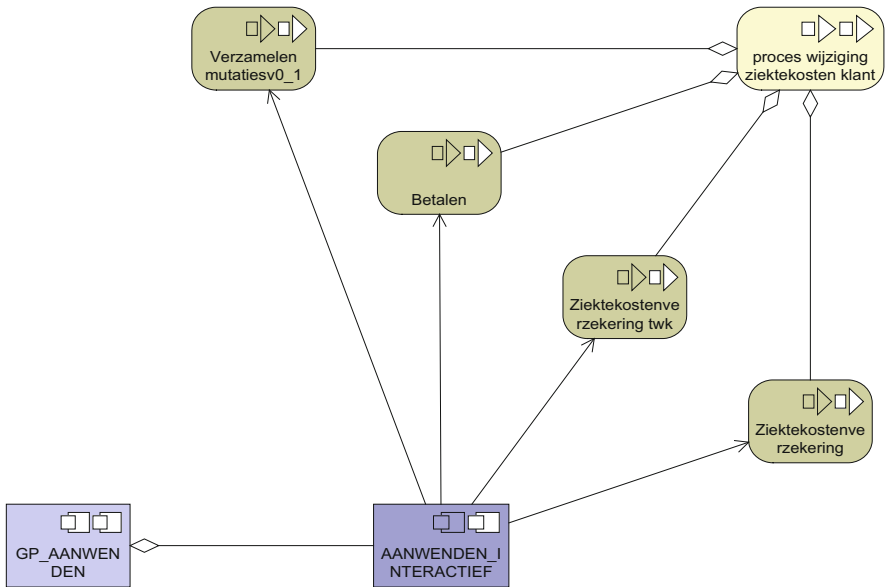


Fig. 12.10 A ‘Process–component relation’ view

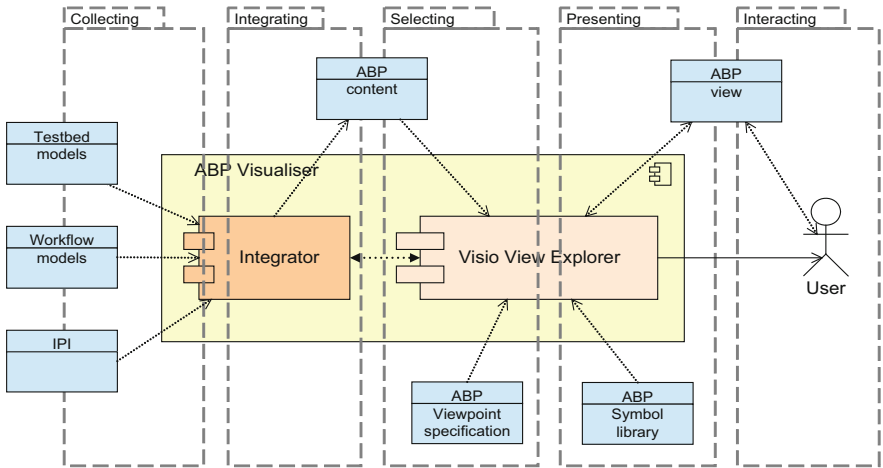


Fig. 12.11 High-level architecture of the ABP Visualiser

integrated data. The data integration itself was done by a technical integration component with data matching algorithms.

Collecting and integrating data are the first two steps of the process. Subsequently, data is selected and presented by the Visio View Explorer on the basis of user specifications. The high-level architecture of the visualiser is depicted in Fig. 12.11. In this figure, the different phases in creating a visualisation are shown. The Visio View Explorer is worked out in Fig. 12.12. The viewpoints are specified in an XML viewpoint configuration file. Together, these two figures are a specialisation of the generic architecture of the ArchiMate workbench shown in Figs. 11.4 and 11.7.

An important part of the generation of visualisations is creating the layout of diagrams. Diagram layout addresses the problem of positioning (possibly nested) boxes and connections on a canvas such that the resulting diagram becomes intuitively acceptable.

In our approach we confined the layout space to a two-dimensional grid. Such an approach is appropriate, because (1) a limited layout space speeds up the layout algorithm, and (2) using a grid causes the resulting diagram to have nicely arranged boxes. Roughly, our diagram layout strategy consists of (1) the estimation of the necessary grid size and (2) the actual positioning of boxes on the grid.

The actual positioning of boxes on the grid was done via a special purpose optimisation strategy.² It generates a layout by minimising the number of crossing connections, box-connection intersections, and the total length of all connections.

²The layout can also be created using general-purpose optimisation technology (e.g., genetic algorithms). We experimented with both the former and the latter, and finally opted for the latter. The main reason for opting for our own optimisation strategy was the problem of level interference: the quality of the visualisation is influenced by the layout of nested boxes.

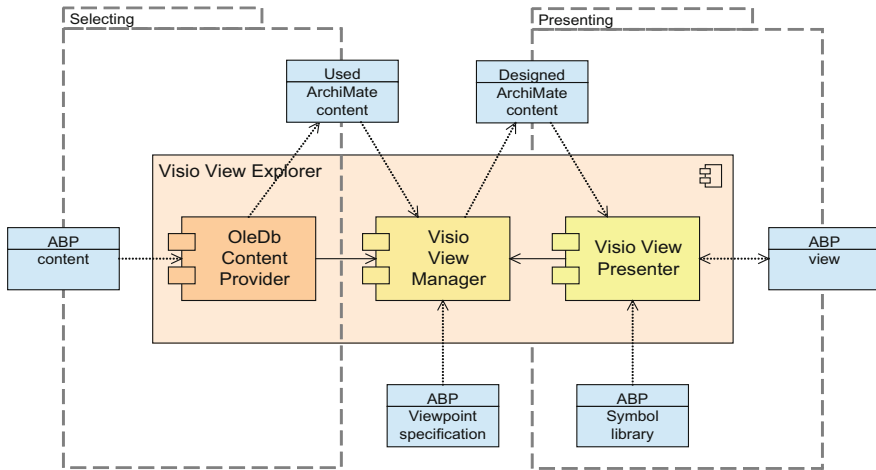


Fig. 12.12 Detailed design of the Visio View Explorer

Furthermore, it is possible to fix boxes in a certain position. In this way we realised a ‘centred’ diagram with one centre box and a ‘flow’ diagram with source and destination boxes.

A typical result is shown in Fig. 12.13: the use of a system component by business processes. A user then can navigate through the visualisations. For example, Fig. 12.13 shows a user zooming in on a particular relation, resulting in Fig. 12.14.

12.1.6 Case Study Results

The results of both case studies were received positively by ABP. Also, ABP’s repository vendor recognised the added value of the case results and enhanced its (newly released) visualisation engine with the insights gained. Via the new repository functionality, ABP’s system owners are now presented with visual representations of the systems for which they are responsible.

12.2 Application Visualisation at ABN AMRO

ABN AMRO is a global bank with a staff of more than 100,000 working in over 3000 branches in more than 60 countries. The bank has a federated, regionally distributed structure with its headquarters in the Netherlands.

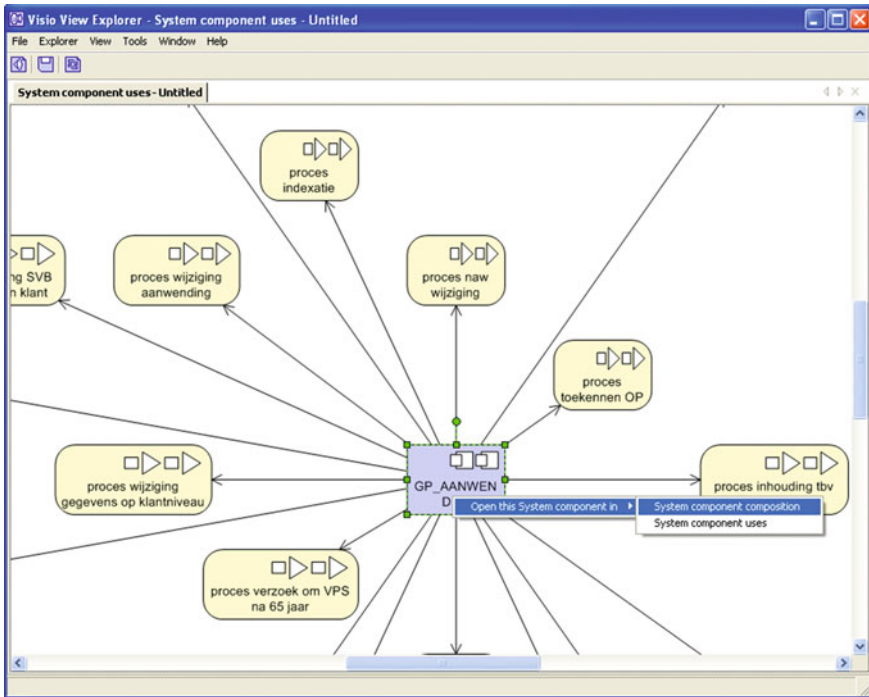


Fig. 12.13 Example: the use of a system component by business processes

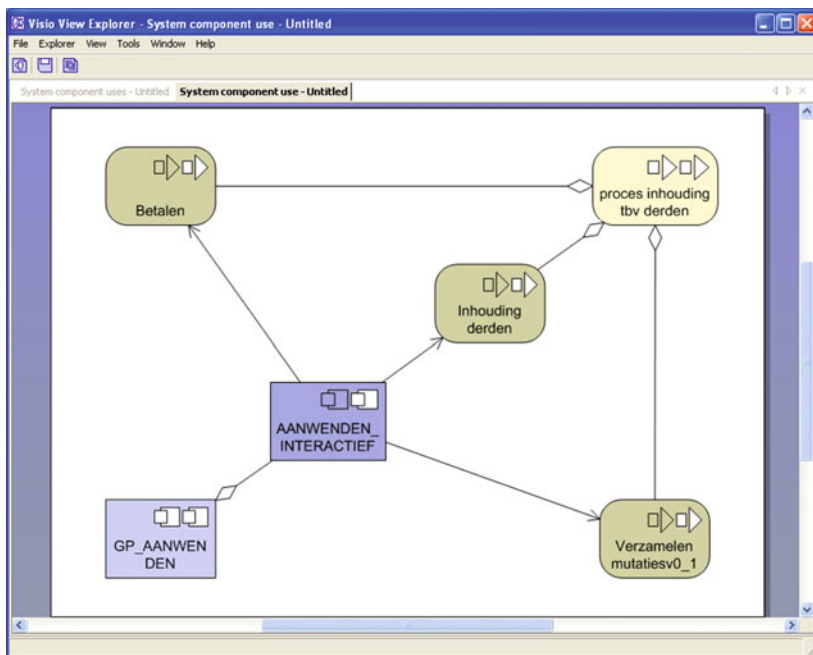


Fig. 12.14 Zooming in on a usage relation between a process and a system

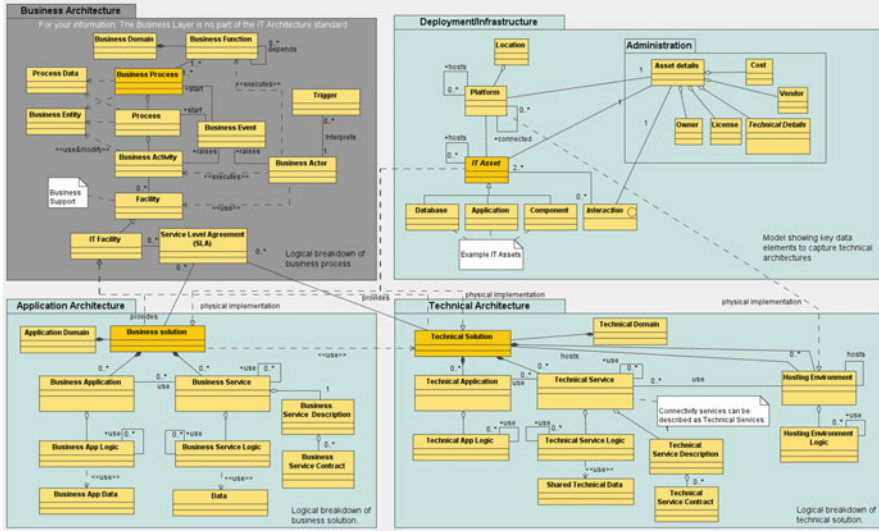


Fig. 12.15 ABN AMRO CITA meta-model

The bank started working with architecture in the middle of the 1990s. In 2000 this resulted in the first version of the Corporate IT Architecture (CITA) method, which primarily defines the organisation of architecture processes and a mandatory set of architecture concepts (in the form of an architecture meta-model). Soon after, a large-scale implementation of this method took place in the Netherlands. The experiences from that implementation and the setup of architecture departments in other countries led to the first set of mandatory corporate policies and standards (P&S) on IT Architecture, called CITA 2003. In 2004 implementation projects started in the USA and Brazil.

One of the P&S of CITA 2003 defines a set of architecture concepts in the form of an architecture meta-model. Another P&S specifies the communication of architecture deliverables, based on viewpoints and views (derived from the IEEE 1471 standard).

12.2.1 CITA Meta-model

The CITA meta-model is shown in Fig. 12.15.³ The overall structure of this meta-model is very similar to the ArchiMate meta-model, with the service concept having a prominent role.

³Actually, the business process quadrant is not yet mandatory.

Table 12.1 Mapping of CITA meta-model to ArchiMate meta-model

CITA concept	ArchiMate concept
Business process	Business process
Business activity	Business activity
Business actor	Business actor
IT facility	Application service
Organisational domain	Grouping relation
Application domain	Grouping relation
Business solution	Application service
Business application	Application component + (External) service
Business service	Application component + (Internal) service
Business application logic	Application function
Business service logic	Application function
Business application data	Data object
(Enterprise) data	Data object

Several differences from ArchiMate are also apparent. For example, although in both meta-models the service concept is used, the meaning is subtly different. In ArchiMate a *service* is a conceptual notion; it does not have to correspond to a particular piece of software. In the CITA meta-model a *service* is an *invokable* piece of external functionality. The ArchiMate *service* concept resembles more the CITA *Business solution* concept and also its counterpart *IT facility*. Finally, the *domain concept* is important to assign domain owners and to group *business solutions* and *technical solutions*. In ArchiMate one would use a *grouping* relation to achieve this.

The case study described here primarily focuses on the business and application architectures. Based on the above explanation, a mapping between the CITA concepts and the ArchiMate concepts is presented for these quadrants in Table 12.1.

12.2.2 Case Essentials

The case study has been carried out in close association with the CITA architecture standard initiative and the work that the Business Unit C&CC (Consumer & Commercial Clients) Brazil, locally known as Banco Real, is doing in the architecture area.

The BU C&CC Brazil is in the process of setting up the architecture profession within the organisation. Currently, it has the following initiatives:

- Introduce the domain architecture function within the organisation.
- Create an application architecture strategy.
- Create a migration plan for this strategy.
- Make an inventory of the ‘as is’ situation from an application architectural point of view.

To support this last effort, BU C&CC Brazil collected information about its information systems using a comprehensive questionnaire. To improve the maintainability and accessibility of this data, a joint effort called CABRI, was set up by the Corporate Centre of ABN AMRO Bank between BU Brazil and ArchiMate. The target for CABRI was to capture essential architectural data from the questionnaire using CITA concept definitions, store them in a database, and generate visualisations based on predefined viewpoints.

12.2.3 Concepts

The CITA standard incorporates the basic principle of a service-oriented environment (SOE) in its meta-model by distinguishing *general-purpose* service components from *specific* business application components. In general this SOE is not yet implemented in full. The goal of applying the CITA meta-model to describe the current state of affairs in the BU Brazil was to identify the gap between the current state and the new application architecture strategy. Several adjustments to the CITA meta-model had to be introduced to show these potential areas for improvement (e.g., reusable functionality and data, ownership of business applications and business services).

Therefore, functionality of systems is split into external usable functionality (*services*) and internal functionality (*application logic*), and the databases are divided into two groups, namely those that contain general-purpose data (*enterprise data*) and those that contain local data (*business application data*). In making this distinction, one has the situation that *business applications* access *enterprise data*, and that *business services* access local *application data*, a situation that the CITA meta-model does not allow. In order not to lose this information, these two access relations need to be added temporarily to the CITA meta-model. The extended subset of the CITA meta-model used in this case study is shown in Fig. 12.16.

In the BU Brazil questionnaire, the primary concepts used were *Systems*, *Macro Functionality*, *Database*, *Domain*, and *Subdomain*. The mapping of these inventory concepts to the extended CITA meta-model is shown in Fig. 12.17. This mapping was used to translate the inventory concepts into CITA concepts, and also to facilitate communication with the Brazilian employees.

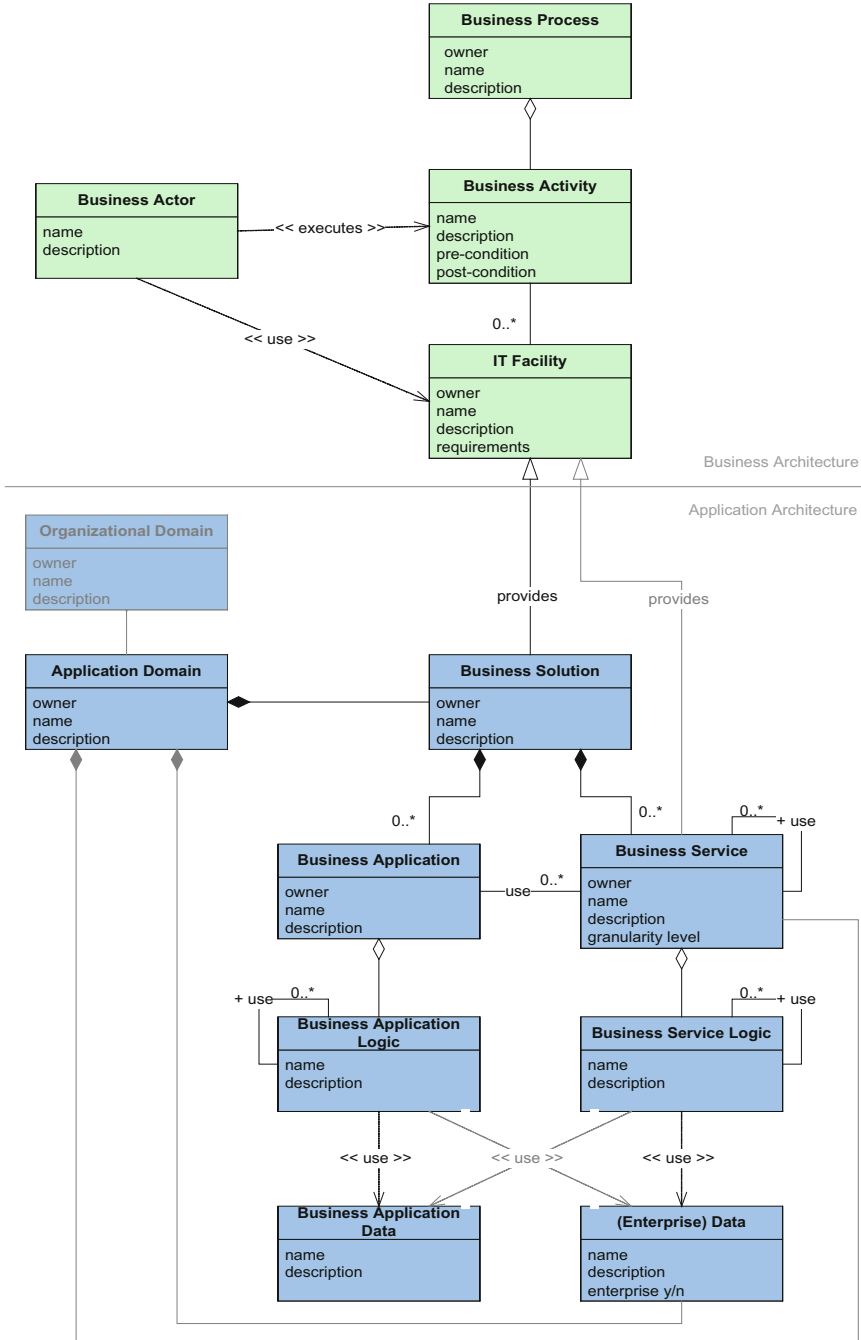


Fig. 12.16 Extended meta-model used in the case study

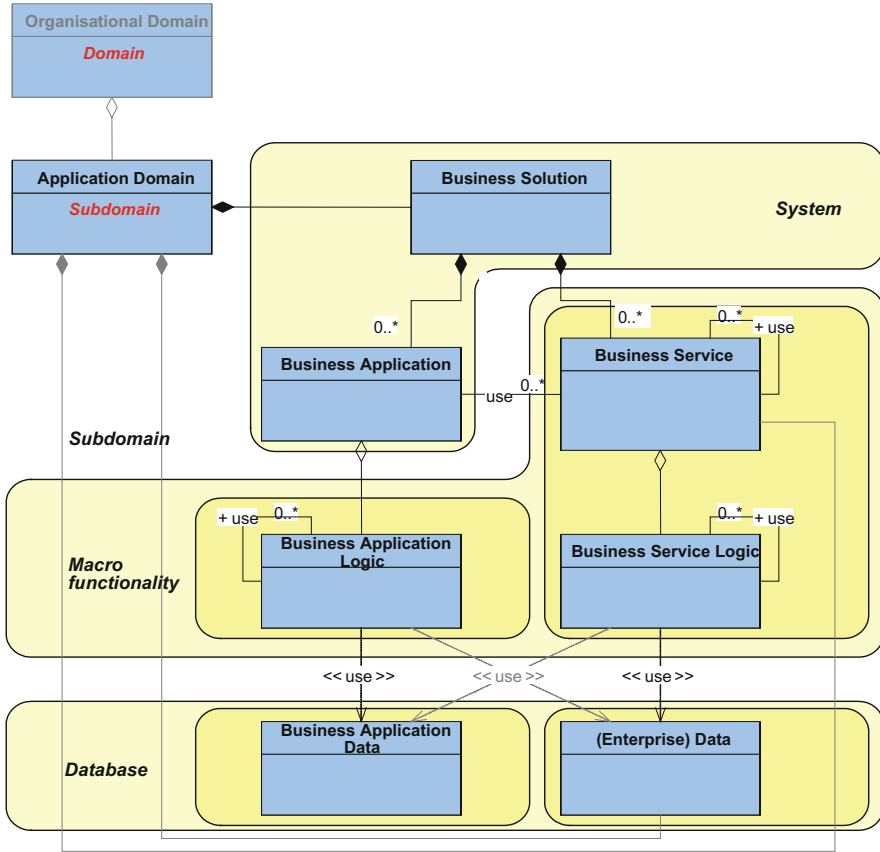


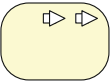
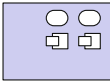
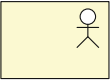
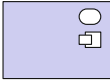

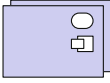

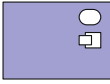



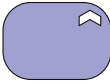
Fig. 12.17 Mapping of inventory concepts to the CABRI meta-model

12.2.4 Visualisation

To visualise the collected architectural information about the IT systems, the following three types of viewpoints have been identified:

1. A global overview of the services and application components (landscape viewpoint).
2. Insight into the support of processes (process support viewpoint).
3. Insight into the relations between applications, services, logic, and data (coherence and dependency viewpoint).

Table 12.2 Concepts and their visual representation

Concept	Symbol	Concept	Symbol
Business process		Business solution	
Business actor		Business application	
Organisational domain		Multiple business applications	
Application domain		Business service	
Business application data		Business application logic	
(Enterprise) data		Business service logic	

The viewpoint description consists of a textual explanation accompanied by an example visualisation. The set of slightly modified ArchiMate symbols used is shown in Table 12.2.⁴

Since the examples used to illustrate these viewpoints in the next subsections are taken from real-life data, some of the text is in Brazilian Portuguese.

Landscape Viewpoints Landscape viewpoints show the overall application architecture, while abstracting from detailed information within these applications. Three different viewpoints have been used: Application domain landscape, Business service landscape, and Business solution landscape.

- The ‘Application domain landscape’ viewpoint shows *Organisational domains* and their containing *Application domains*. This viewpoint is mainly concerned with visualising (levels of) ownership (Fig. 12.18).
- The ‘Business service landscape’ viewpoint shows one *organisational domain*, with all its *application domains*, and all their *business services* (Fig. 12.19).
- The ‘Business solution landscape’ shows one *organisational domain*, with all its *application domains*, with all their *business solutions* (not shown here, but analogous to the previous figures).

⁴The concepts for business activity and IT facility are not used in these visualisations.

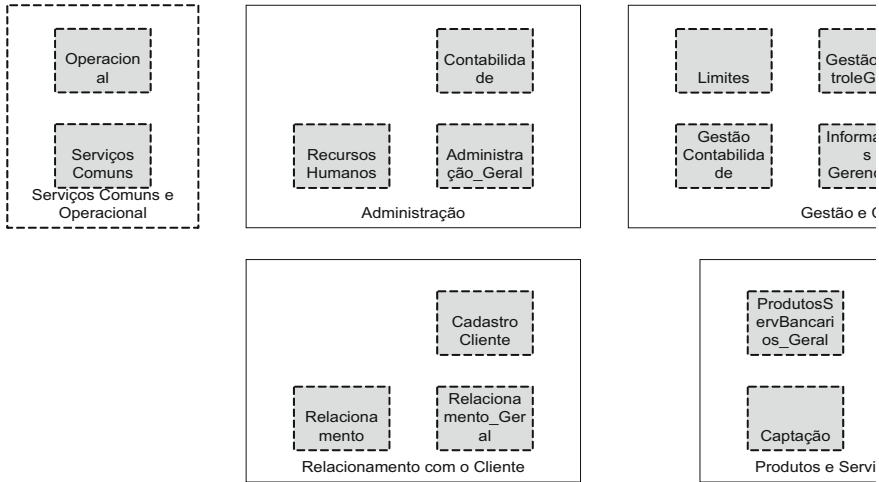


Fig. 12.18 An 'Application domain landscape' view (partial)

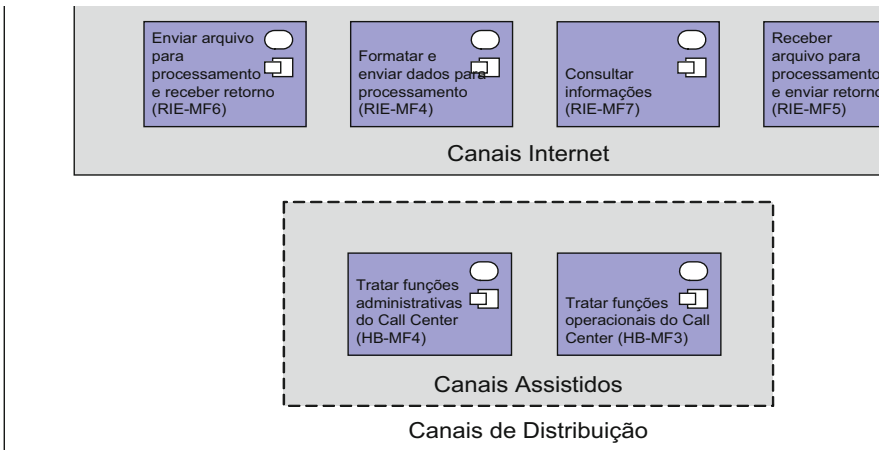


Fig. 12.19 A 'Business service landscape' view (partial)

Process Support Viewpoints Process support viewpoints facilitate insight into the relation between processes and applications.

Currently, only one viewpoint has been identified and worked out: the 'Business activity–business service alignment' viewpoint. This viewpoint shows one central *Business process*, together with the *Business activities* involved in that *Business process*. Each *Business activity* depicts the *Business services* that are used by it (Fig. 12.20).

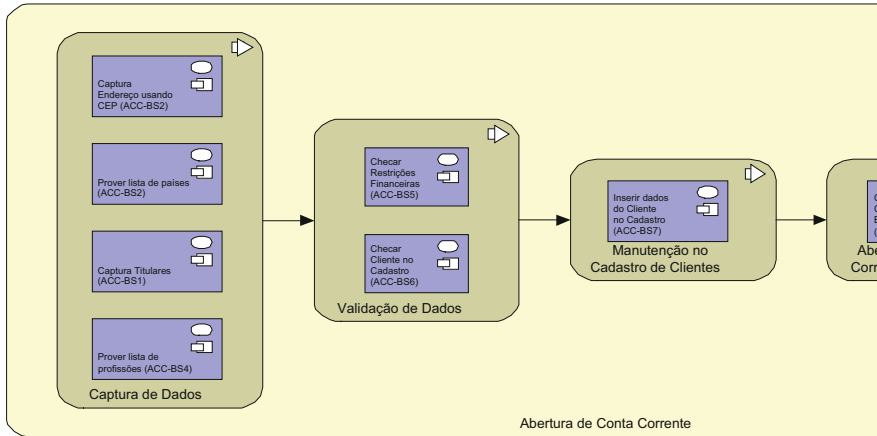


Fig. 12.20 A ‘Business activity–business service alignment’ view (partial)

Coherence Viewpoints Coherence viewpoints facilitate insight into the coherence of the application architecture. They show how a particular element is used by other elements. Three coherence viewpoints have been worked out: Business service uses, Business application data uses, (Enterprise) data uses.

- The ‘Business service usage’ viewpoint shows one central *business service* surrounded by the *business applications* and *business actors* that use it (Fig. 12.21).
- The ‘Business application data usage’ viewpoint shows one central *business application data* entity surrounded by the *business application logic* entities and *business service logic* entities that use it (Fig. 12.22).
- The ‘(Enterprise) data usage’ viewpoint shows one central (*enterprise*) *data* entity surrounded by the *Business application logic* entities and *Business service logic* entities that use the central data entity (not shown here, but analogous to Fig. 12.22).

Dependency Viewpoints Dependency viewpoints facilitate insight into the dependencies of the application architecture. They show a central entity together with certain entities on which this central entity depends. The following viewpoints have been identified: Business application dependencies, Business application logic dependencies, and Business service dependencies.

- The ‘Business application dependencies’ viewpoint shows one central *Business application* surrounded by the *Business services* that are used by that central *Business application* (Fig. 12.23).
- The ‘Business application logic dependencies’ viewpoint shows one central *Business application logic* entity surrounded by the *Business application logic* entities, *Business application data* entities and (*Enterprise*) *data* entities that are used by the central entity (analogous to Fig. 12.23).

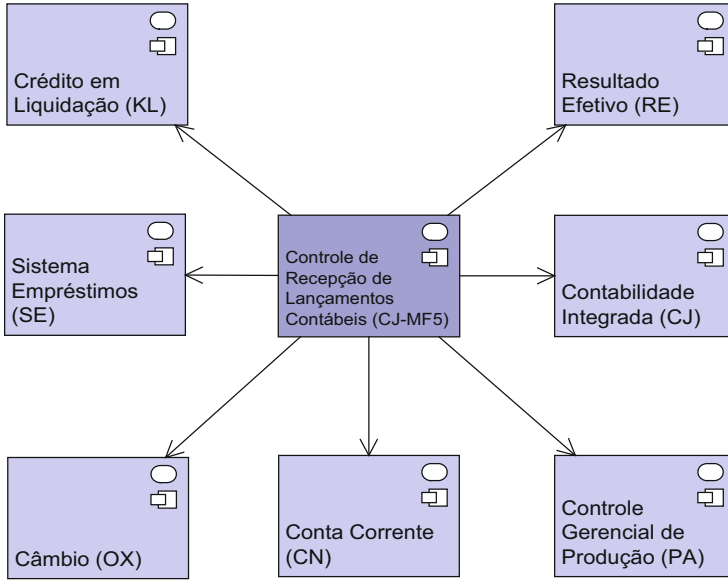


Fig. 12.21 A ‘Business service usage’ view

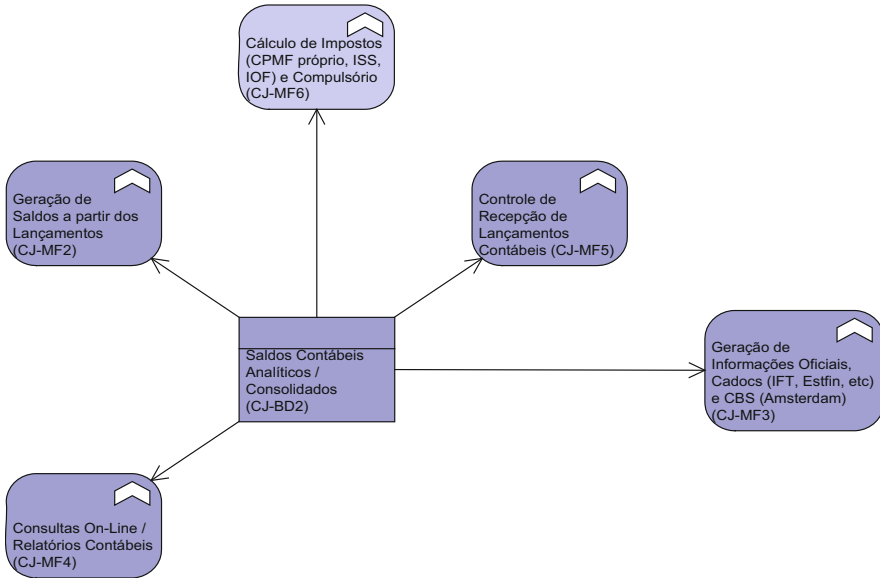


Fig. 12.22 A ‘Business application data usage’ view

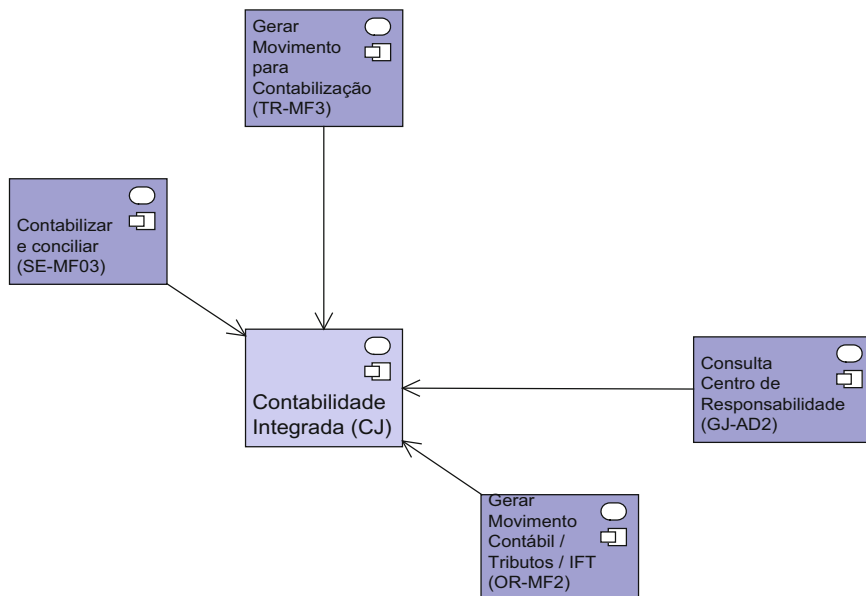


Fig. 12.23 A ‘Business application dependencies’ view

- The ‘Business service dependencies’ viewpoint shows one central *Business service* entity surrounded by the *Business service* entities, *Business application data* entities and (*Enterprise*) *data* entities that are used by the central entity (analogous to Fig. 12.23).

12.2.5 Tool Design and Results

As in the case of ABP, the ABN AMRO case study uses the generic tooling infrastructure described in Chap. 11. The way the tool is used in this case is shown in Fig. 12.24. Input of system information is not yet automated, since this is only available in the form of textual documents.

The practical results obtained with this visualisation infrastructure helped to clarify various misunderstandings and inconsistencies in the systems landscape. The visualisations are widely and interactively used in discussions about the current and future application architecture.

ABN AMRO BU Brazil decided to keep using the tool and implement it in the development organisation. It will capture systems that have not yet been assessed and will maintain the data already captured.

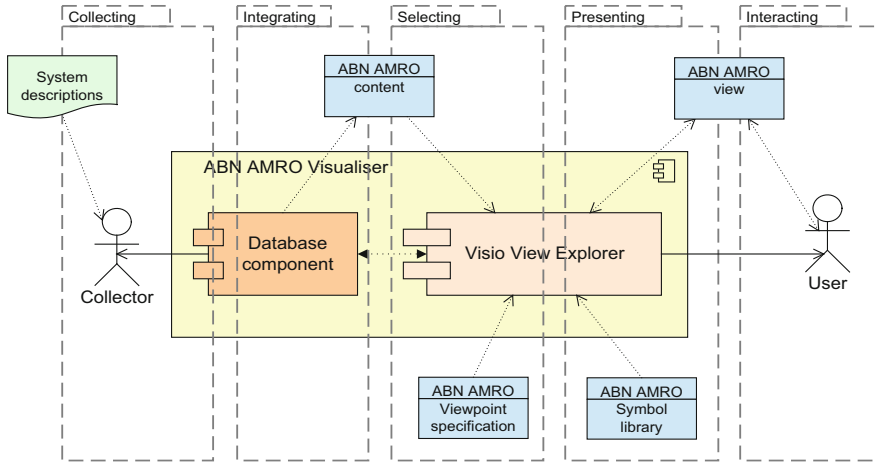


Fig. 12.24 Tool design

12.3 Design and Analysis at the Dutch Tax and Customs Administration

The Dutch Tax and Customs Administration (abbreviated TCA in the sequel) has a long history of continuously improving its organisation of process and ICT development. As early as the beginning of the 1980s, the ICT department started working with architecture. In the TCA architecture plays a prominent role, which is also exemplified by a total staff of over 100 architects. The importance of architecture has also increased the need for an enterprise architecture language to connect different architecture domains.

12.3.1 Case Essentials

In recent years, the organisation of social security in the Netherlands has changed dramatically. The goal is to arrive at a situation with a central contact point for organisations and citizens, and with unique ‘authentic’ data sources.

Within this context, the collection of employees’ social security premiums is transferred from UWV (the central social security organisation) to the TCA. This joint project of TCA and UWV is called SUB (‘Samenwerking UWV–Belastingdienst’).

A major challenge in this project is to handle enormous flows of data within and among the different organisations. This concerns more than 600,000 payroll tax returns each month, a large proportion of which arrive within a peak period of a couple of days. Moreover, it is expected that a substantial proportion of these tax

returns need to be sent back for correction. Such requirements need to be addressed early on in the project.

These aspects of this case study made it an ideal proving ground for the modelling language, viewpoints, and performance analysis techniques described in previous chapters. In the next subsections, we will show how the different aspects of the business processes, applications, and infrastructure were modelled in a coherent and consistent way, and also show how the quantitative analysis techniques were used in the capacity planning of the infrastructure.

12.3.2 Views

By means of a number of different views, based on the design viewpoints described in Sect. 8.5, the SUB information system architecture is presented from the perspective of the TCA. We have chosen not to show a model of SUB as a whole; instead, we start with a broad perspective and go into detail for a number of specific processes.

Subsequently, models are presented that describe the SUB business processes (viewpoint Process cooperation), the SUB application support for these processes (viewpoint Application usage), and the infrastructure support for the applications (viewpoint Infrastructure support).⁵

Process Cooperation: Client-to-Client Processes The process architecture, depicted in Fig. 12.25, shows the most important client-to-client processes within the scope of SUB. Each process is initiated by a *trigger*. These triggers fall in one of the following categories:

- *time* triggers, indicating that a process is executed periodically;
- *message* triggers, indicating that an incoming message initiates a process;
- *signal* triggers, indicating that an incoming signal initiates a process.

For each trigger, a *frequency* is specified, expressed in terms of the average number of ‘firings’ per month. Furthermore, the process architecture shows the most important messages that flow between the processes.

Obviously, each of the above-mentioned client-to-client processes can be described in more detail by further specifying the sub-processes of which they consist, the actors that are involved, the incoming and outgoing messages and the databases that are being used. Next, we present a more detailed decomposition of the process ‘Payroll tax return’ (Fig. 12.26) from the overall SUB process architecture. The model shows, among others, which part of the process is executed by the TCA and which by UWV.

⁵The actual design of SUB further evolved after completion of the case study.

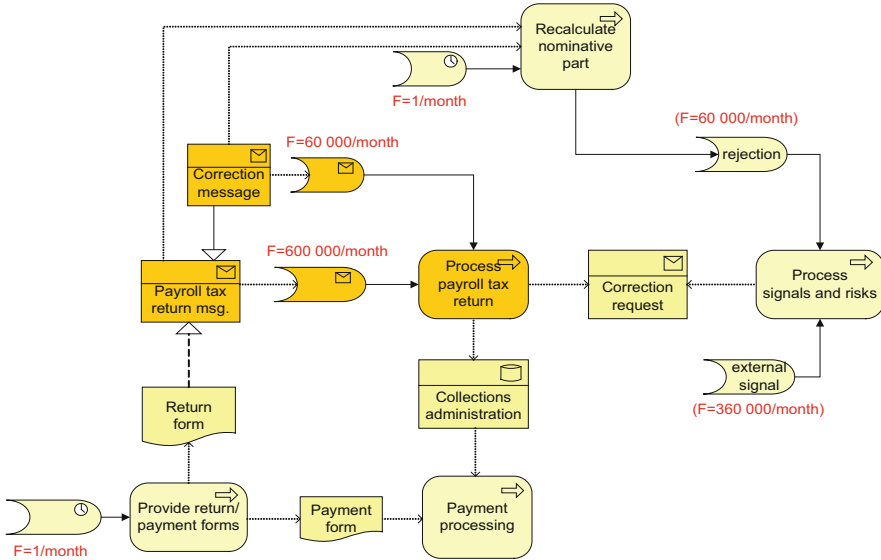


Fig. 12.25 Overview of the SUB client-to-client processes

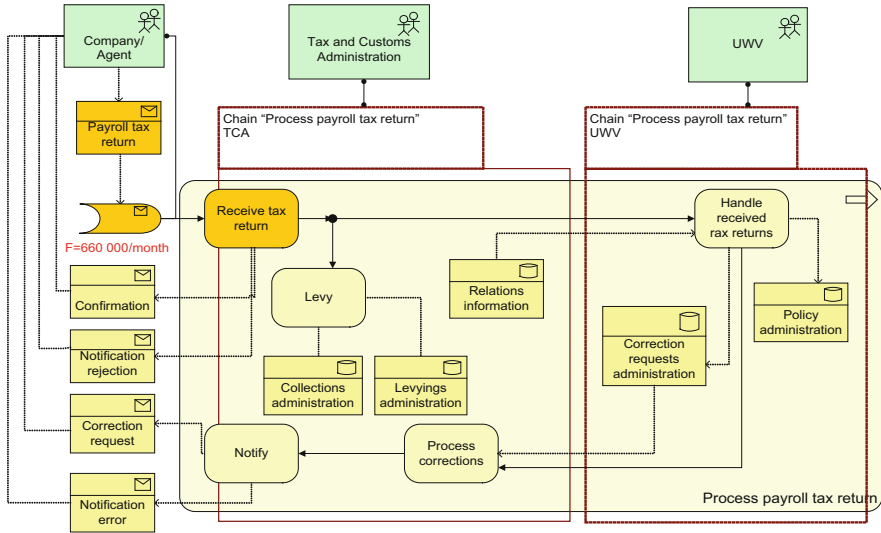


Fig. 12.26 Client-to-client process ‘Payroll tax return’

Application Usage Going one level of detail deeper, we now zoom in on the ‘Receive tax return’ sub-process. The model of this sub-process and the corresponding application support are shown in Fig. 12.27. A payroll tax return (PTR) can be submitted in two main formats: on paper or electronically. The

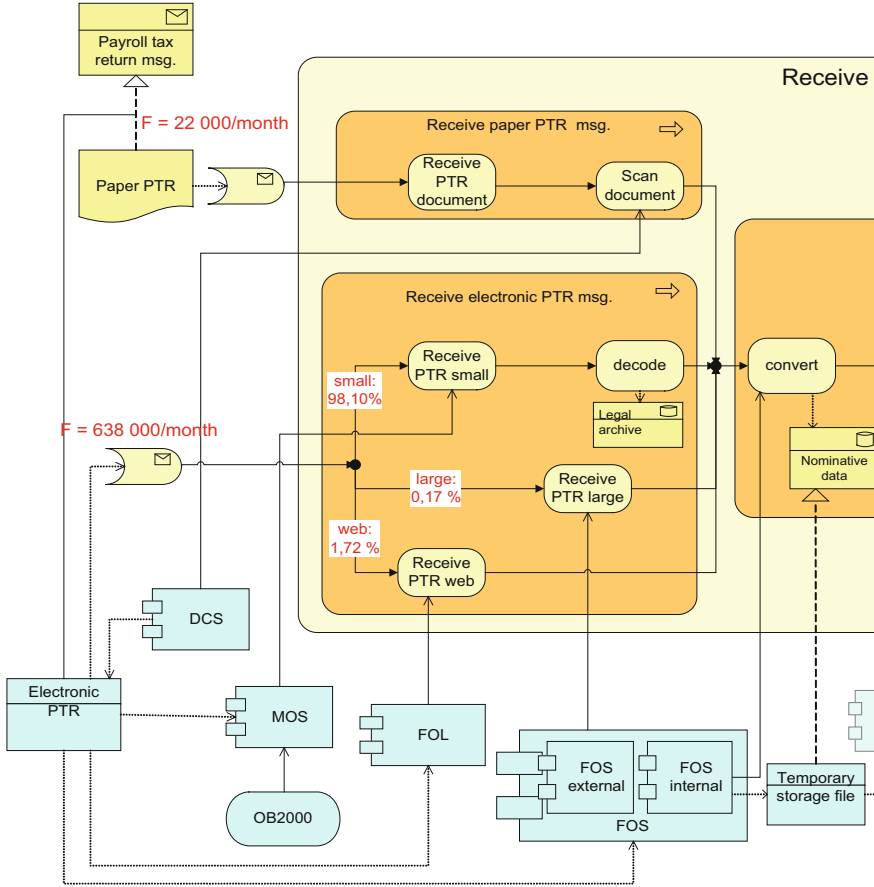


Fig. 12.27 Application and business process architecture for ‘Receive tax return’ (partial view)

electronic tax returns have three possible formats: Web-based messages, small messages sent via SMTP, and large messages sent via FTP. The model shows the expected distribution of the total number of messages over these different formats. The first part of the ‘Receive tax return’ process transforms these formats into a common, medium-independent format. We will refer to this phase of the process as ‘Medium-specific processing’. The second phase of the ‘Receive tax return’ process, ‘Medium-independent processing’, processes all the payroll tax returns in the same way, irrespective of their original format.⁶

⁶The applications shown in Fig. 12.27 with a lighter colour, i.e. BvR, BBA, WCA and Notification, are mainly databases that are used in the processes, but play a secondary role. They are omitted in the more detailed models and the analysis.

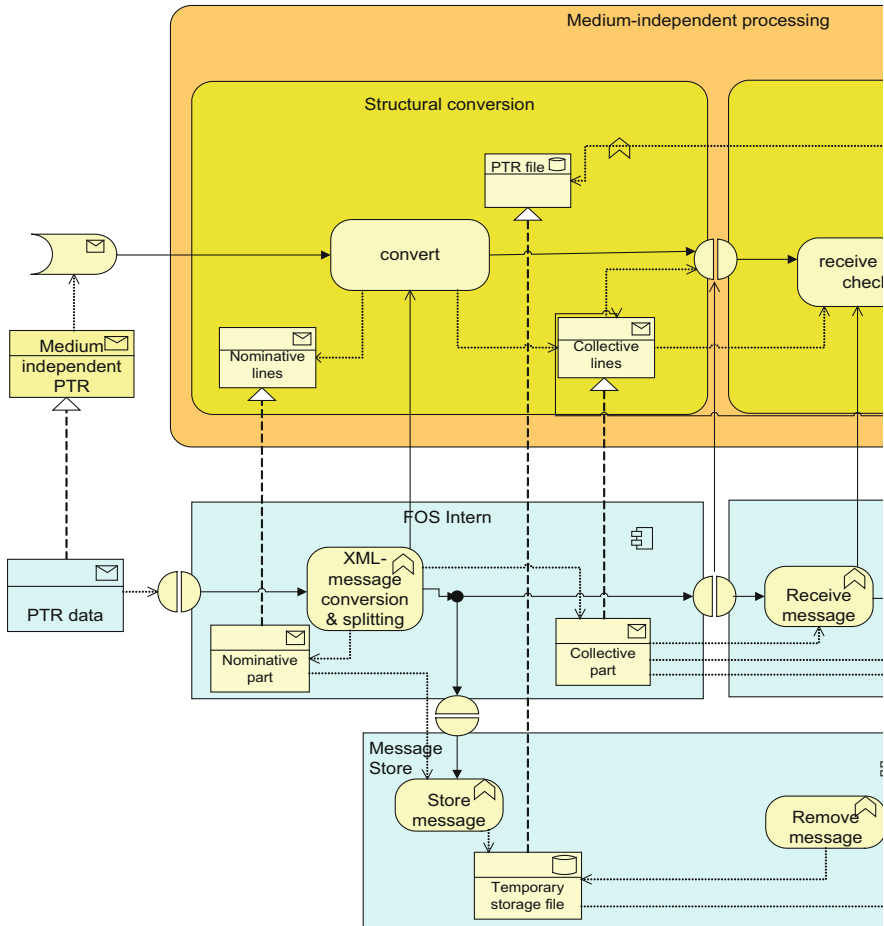


Fig. 12.28 Application support for ‘Medium-independent processing’ (partial view)

First, we detail the ‘Medium-independent processing’ phase. In the application architecture, the behaviour of each application component is partitioned into one or more *application functions* (denoting units of functionality used within the business processes) and *application interactions* to model communication between application components, as well as the data stores involved. Part of the resulting model is shown in Fig. 12.28.

Infrastructure Usage The next step is to take a closer look at the infrastructure support for the application architecture. We first illustrate the modelling approach for the ‘Medium-independent processing’. A layer of infrastructure services supports the various application functions. We distinguish three types of infrastructural services:

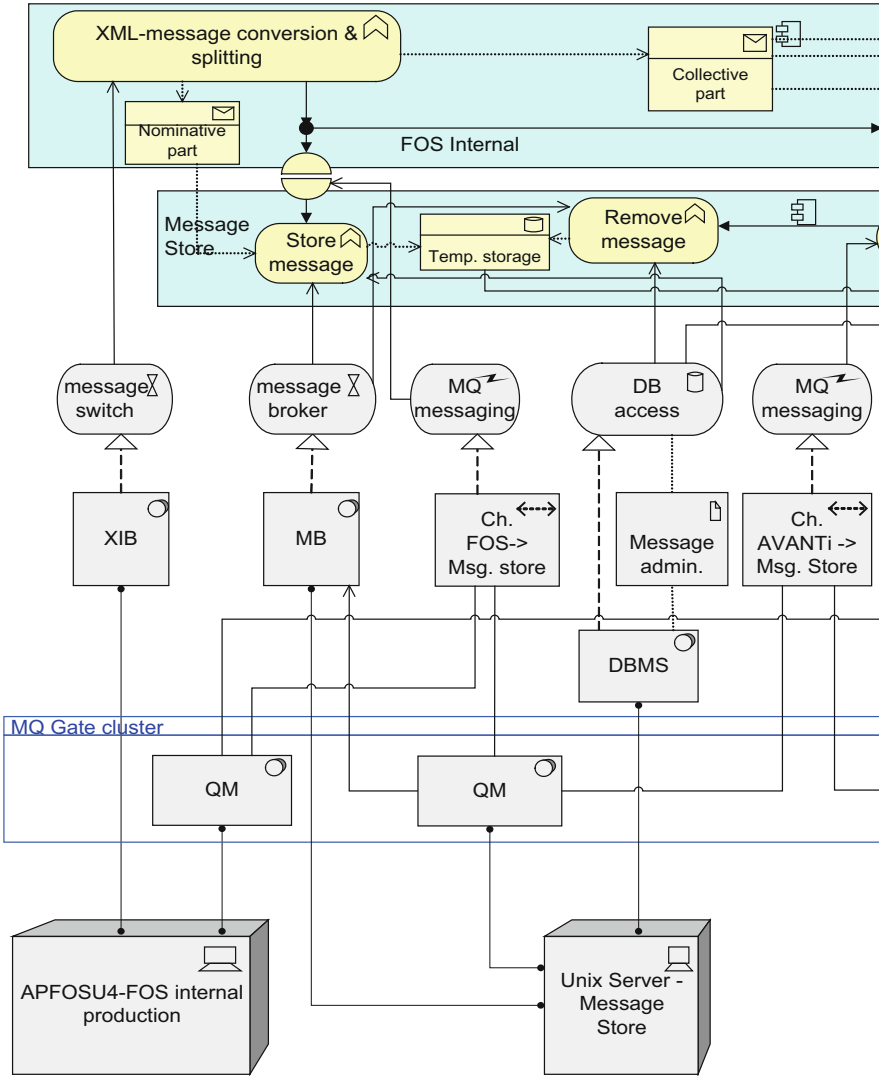


Fig. 12.29 Application and infrastructure architecture for ‘Medium independent processing’ (partial view)

- data storage and access services;
- processing services;
- communication services.

Data storage and access services are realised by, for example, a database management system. Processing services are typically realised by an execution environment or application server. Communication services support messaging between applications which is realised by, for instance, message queuing software (Fig. 12.29).

In this case, WebSphere MQ technology is used, where message brokers and message switches make use of functionality provided by queue managers. In MQ, communication services are realised by so-called *channels*. A channel between two devices is modelled as a communication path that represents a collaboration of two QM system software components, one for the sender and one for the receiver.

As mentioned above, the first part of the ‘Return tax returns’ process, ‘Medium-specific processing’, receives payroll tax returns from four information sources. Following the same modelling guidelines as in the case of the ‘Medium-specific processing’ part, we present in Fig. 12.30 the whole layered architecture

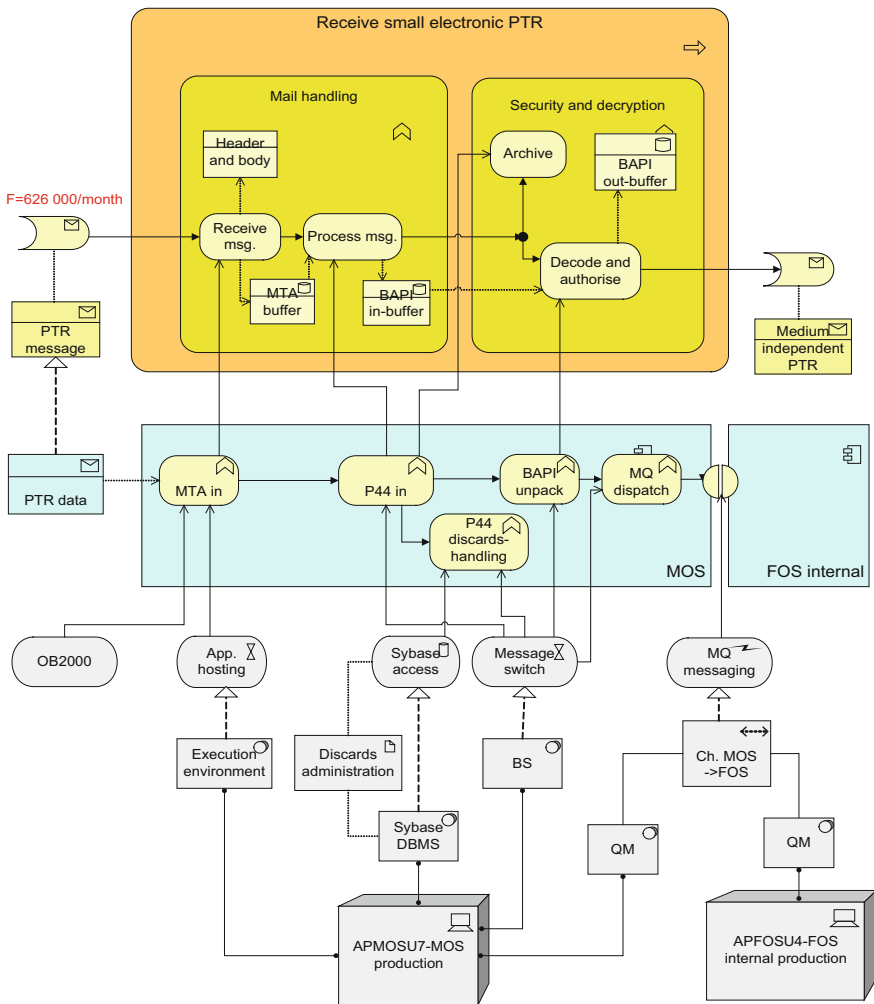


Fig. 12.30 ‘Receiving small electronic payroll tax returns’ architecture

(i.e. business process, application and infrastructure architecture) of ‘Receiving small electronic payroll tax returns’.

The models for the other three sources of tax returns will not be shown here, but they can be constructed in a similar way.

Infrastructure Support So far, we have adopted a top-down approach: starting with the business processes, we first identified the needed application support; then, we specified the infrastructure needed to run the applications. In this view, we work bottom-up: we show the complete infrastructure within the scope of the ‘Receive tax return’ process for SUB, and show which of the infrastructure services are used by which of the applications. Part of this view for the ‘Receive tax return’ process is shown in Fig. 12.31.

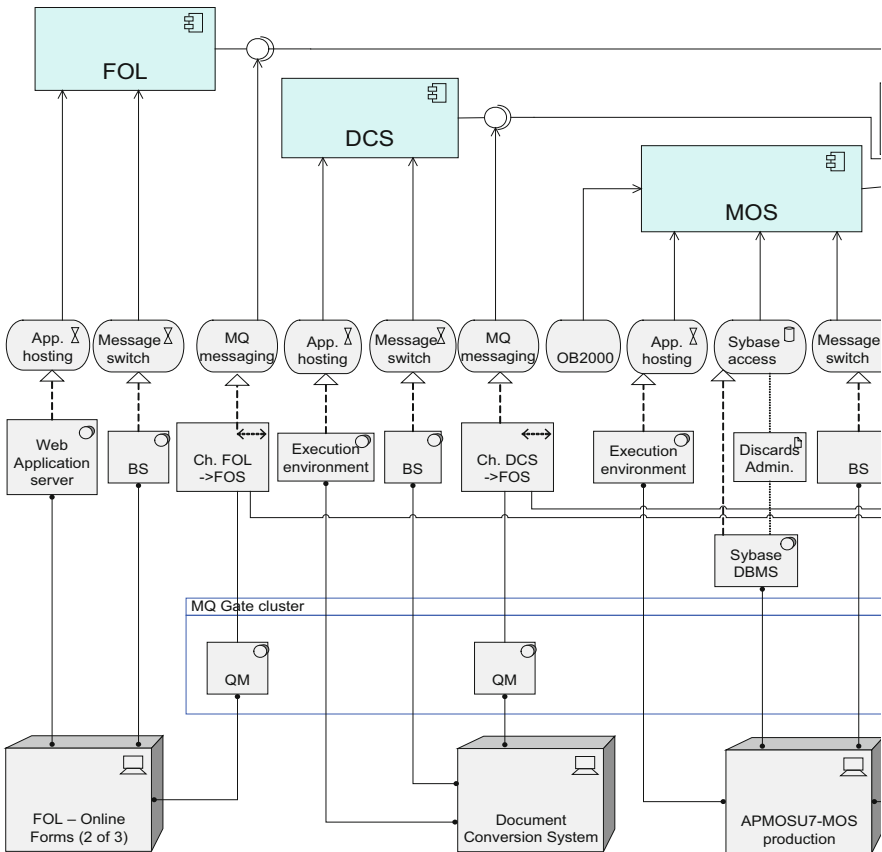


Fig. 12.31 Part of the SUB infrastructure support for applications (partial view)

12.3.3 Performance Analysis

This section illustrates the quantitative analysis of the model presented in the views in the previous sections, using the analysis approach described in Chap. 9. The results can be used to get an indication of the capacity that is required for the different resources in the infrastructure layer.

Analysis Approach For the given type of analysis, the following input data is required:

- For each trigger the *arrival frequency* (average and possibly also peaks).
- For each process, function, or service the average *service time*.
- For each actor, component, or device the *capacity*.

Given these inputs, we can estimate the following performance measures:

- For each concept in the model (service, process, function, and resource) the *throughput*: the number of inputs/outputs that is to be processed per time unit. This is the *workload* that is imposed by the processes.
- For each actor, component, and device its *utilisation*: the percentage of time that it is active.
- For each process, function, and service the average *processing time* and *response time*.
- For each client-to-client process the average *completion time*.

The analysis approach is portrayed in Fig. 12.32. Starting with the arrival frequencies on the business process level, the workload (throughput) for all model elements in the layers below is calculated (top-down analysis). Together with the given service time of the infrastructure services, the utilisation of the resources, and the processing and response times of the processes, functions, and services are calculated (bottom-up analysis). In Sect. 9.2 there is a detailed description of the analysis algorithms.

Workload Calculations (Top-Down) Some of the results of the workload calculations are shown in Fig. 12.33 (in italics). These figures reflect the workload of applications and infrastructure imposed by the sub-process *Medium-independent*

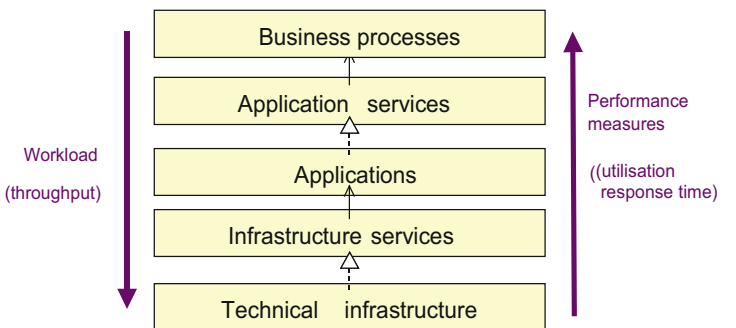


Fig. 12.32 Overview of the analysis approach

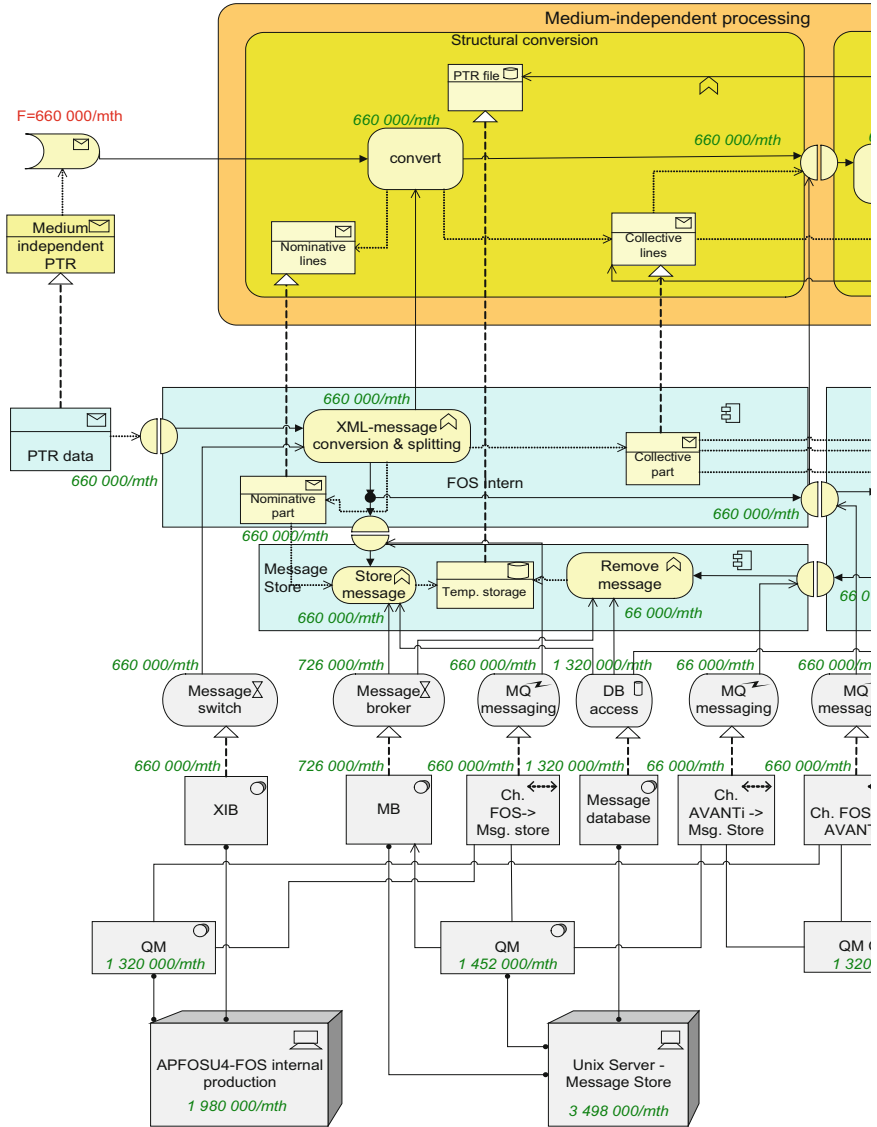


Fig. 12.33 Throughputs for the sub-process ‘Medium-independent processing’ (partial view)

processing, given an average monthly supply of 660,000 payroll tax returns. This workload is the basis for further performance analysis.

To obtain estimates of the total required infrastructure capacities, the same calculations also have to be made for the different *Medium-specific processing* parts of the *Receive tax return* process. The sum of the workloads from all the sub-processes results in a total workload for the SUB infrastructure, part of which is shown in Fig. 12.34. Similar calculations could be carried out for peak situations.

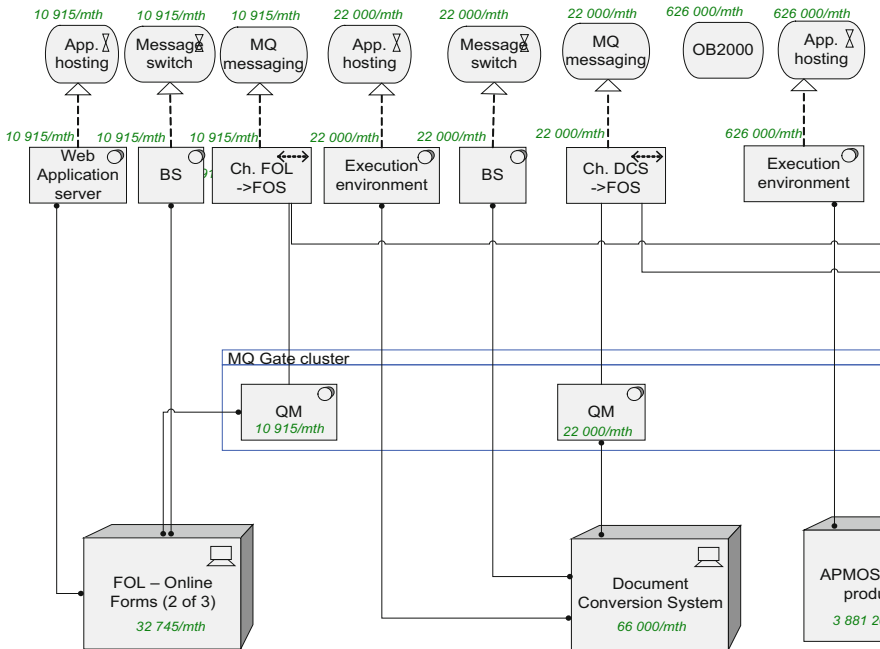


Fig. 12.34 Total workload of the SUB infrastructure (partial view)

Performance Measure Calculations (Bottom-Up) To calculate performance measures such as response times and utilisation, service times are also needed as input data. These figures are often difficult to establish, especially in a design phase of a project when systems are not yet operational. Nevertheless, based on technical documentation and available historical information (e.g., performance tests) of existing system components, and together with experts on the matter, reasonable estimates of these numbers could be made.

The numerical results of the bottom-up analysis of the process ‘Receiving small electronic payroll tax returns’ are given in Fig. 12.35. According to these figures the utilisation of the resources for an average workload is already quite high; this means that at peak loads the resources will almost certainly be overloaded. A solution to this problem may be to add additional resources or to increase the capacity of the resources. Further analysis can help to determine by how much the capacity needs to be increased.

12.3.4 Case Study Results

This case study shows that the ArchiMate language is suitable for modelling the relevant aspects of the technical architecture, as well as the relations of this

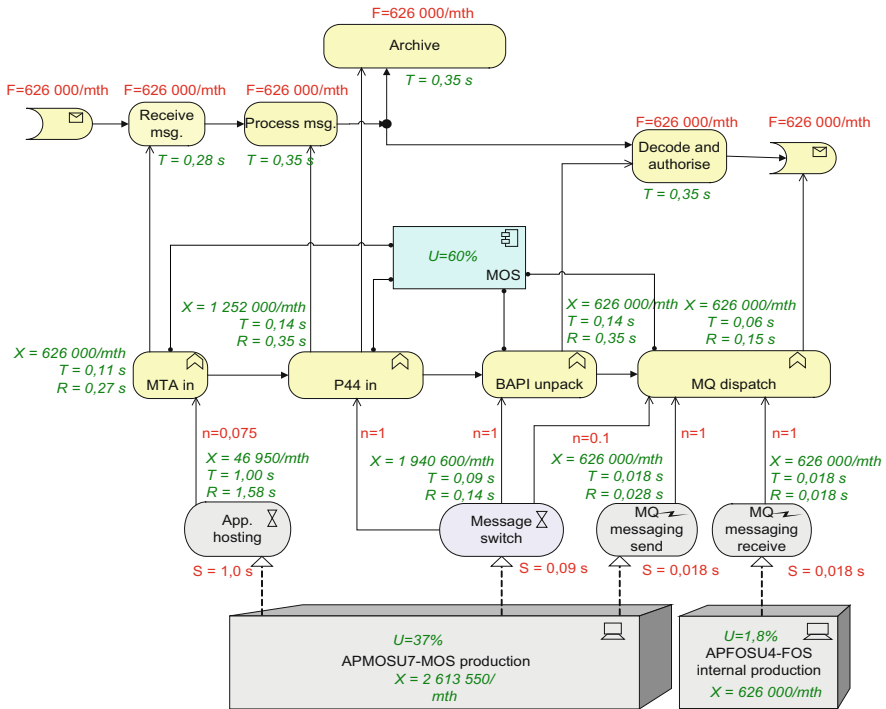


Fig. 12.35 Utilisation and response times for ‘Receive tax returns (small)’

architecture to other architectures. The resulting models make the realisation of generic infrastructure services explicit. Quantitative analysis offered a clear view of how activities at the business process impose a workload on the application and infrastructure levels, thus providing a basis for capacity planning of the infrastructure. Performing these quantitative analyses at an early stage, considerably helps the realisation of the desired performance characteristics of the target system.

12.4 Summary

The case studies discussed in the previous sections represent only a small part of all the applications and validations of the methods and techniques presented in this book. However, they clearly show the feasibility and practical value of these results in various real-life settings. Both the modelling language and the visualisation and analysis techniques have shown their merit in providing more insight into complex, wide-ranging enterprise architectures.