Chapter 11
Guidelines for the Specification of New Administrative Processes

This chapter presents some guidelines on how to specify a new administrative process (*composite external service* in the terminology of Chap. 10) in the eG4M methodology on the assumption of using a service-oriented architecture (SOA) and related engineering approaches, techniques, and tools. In particular, the guidelines address the case in which an external service has been identified as target of the project (cf. Chap. 9) and the realization of such a service requires the coordination of different PAs, which have to offer (or already offer) inter-PA internal services (cf. Chap. 10) to be coordinated. Before moving to the effective realization of the project (out of the scope of this book), a final step of operational planning is needed to analyze and conceptually model the new services and the coordinating process, which serve as technical input for possible tenders aimed at effective realization. Section 11.1 presents the overall description of the guidelines, by introducing the steps to be realized in order to automate the process. Then Sect. 11.2 outlines the modeling tools, i.e., the techniques and the possible software tools that we suggest to use in order to support the managers and system/software engineers during the specification of the process at design-time. Section 11.3 gives some highlights on the important issue of managing legacy systems. Finally, Sect. 11.4 outlines, through a specific case study, how to specify an entire process. This chapter assumes the reference architecture and the eGovernment cooperative support system as presented in Chap. 10, as they are the target deployment environment for the artifacts produced by the guidelines presented here.

11.1 Overview of the Guidelines

The presented guidelines aim to specify the automation of macro-processes (see Chap. 5) among different PAs in order to implement more efficient and advanced external composite services toward the clients requiring such services. The guidelines aim toward the realization of three objectives:

This chapter is authored by Massimo Mecella.

G. Viscusi et al., *Information Systems for eGovernment*,
DOI 10.1007/978-3-642-13571-2_11, © Springer-Verlag Berlin Heidelberg 2010

- *Formalization of the process*: Assuming that the service to be automated has been previously identified (cf. Chap. 9), it consists in
  - the conceptual description of the process realizing the service through appropriate modeling tools, namely business process modeling notation – BPMN – cf. Appendix B and unified modeling language – UML or entity – relationship model;
  - the description of operations enacted by such a process through suitable technological languages (e.g., XML, WSDL, WS-BPEL); see Chap. 10;
- *Definition of the specific cooperation domain*: It consists in the precise description of different services involved in the process and of the data exchanged among PAs.
- *Realization of the process/service*.

Overall the guidelines consist of five activities, as shown in Fig. 11.1. We now comment the activities:

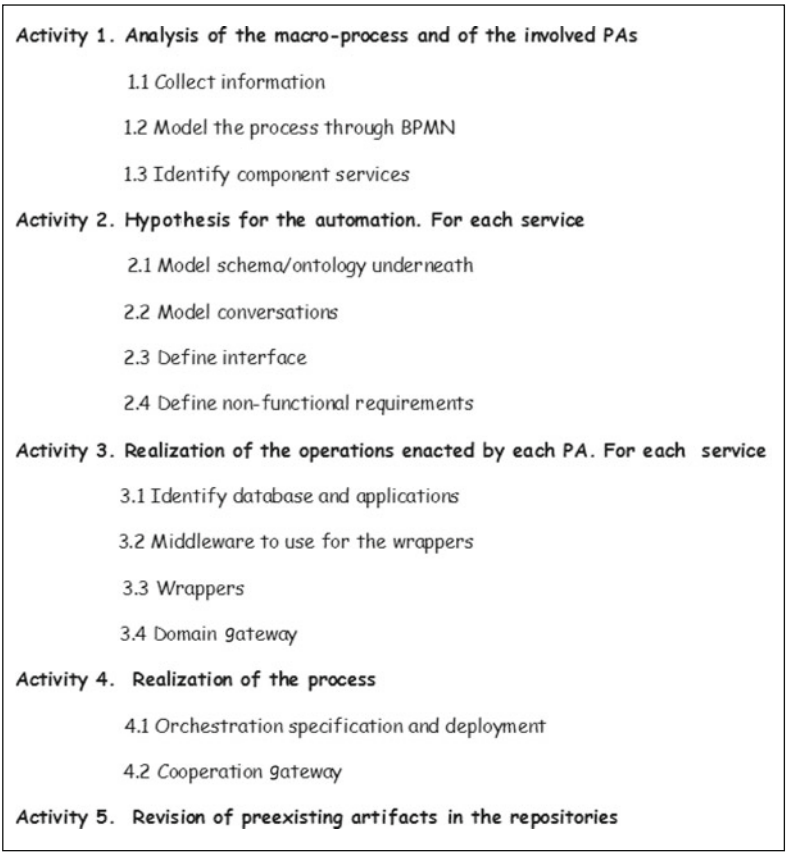

Fig. 11.1 The activities of guidelines at first glance

1. *Analysis of the macro-process and of the involved organizations*: It is assumed that the macro-process to be automated has been identified, as discussed in previous chapters. It is important to note that automation should aim to provide a more efficient external service to clients by improving the quality (see Chap. 7) reducing the delays. Delays are due to inefficient interoperation among different PAs involved, due to differences in the information systems, to missing information, and to the fact that currently most information exchanges require using hard-copy documents sent through ordinary mail with the client who often serves as a messenger, providing the information needed to establish communication between the PAs. The choice of the process, as discussed previously in the book, is based on the usefulness, the political/strategic visibility of the process, and the technological readiness of the clients in the use of new services. Indeed, according to what was discussed in Chap. 10, the new service will be an *e*-service, and therefore, if the target users are not ready to use electronic channels, the automation may be hindered or reduced. Examples may be health services for elderly persons (i.e., not customary with the use of Internet and WWW browsers, etc.), which may be automated, but still need to be provided through a classical front-desk, and not only as an *e*-service on the Web.
   The analysis of the chosen macro-process starts by collecting some information needed for the subsequent specification, namely

   - number and type of clients using the process;
   - the description of the old process;
   - documents and information related to the pre-existing component services offered by the PAs involved (e.g., documents describing the service, documents describing the laws that regulate the supply of the specific service, the identification of PAs which are in charge of supplying the service, the description of the internal administrative procedures currently in use in the PAs);
   - the legacy information systems of such PAs.

   All of the above information should have been collected during the state reconstruction step (see Chap. 5). It could be necessary at this stage to complete the documentation for specific or more detailed aspects. This way, the relationships among (i) process/services, (ii) PAs, and (iii) data exchanged are clearly and completely identified. This activity requires the participation of the staff (technicians, civil servants, and managers) of the PAs involved. Having all this information, this activity consists in the analysis and modeling of the macro-process.

   In particular, the use of the language BPMN is suggested in order to represent the workflow of the process and to model parallel processes and their synchronization. Moreover, it is suggested to use UML sequence diagrams to specify the dynamics of the system and the interaction among the subjects involved, showing the temporal sequence of the messages exchanged by them. The language BPMN is described in Appendix B, whereas for a comprehensive introduction to UML and UML diagrams, see [159].

2. *Hypothesis for the automation of the processes*: The previously elaborated models representing the logic of the process are used as the starting point for the definition of the component services, namely inter-PA internal services. Here the following should be defined:

- The conceptual schema/ontology underneath the component service: it is necessary because in dynamic and decentralized contexts, each subject involved has to define suitable schemas/ontologies supporting the comprehension of the semantics of the service and of information carried by it.
- The service interface: it is defined as the set of operations (each one with its own signature) offered by the service.
- The service conversational protocol: it describes which sequences of operations (conversations) are supported/admitted by the service.
- A set of non-functional requirements: they express the levels of quality for the provided service, the characteristics of security of the service, etc.

Basically, this activity identifies the conceptual elements needed in order to implement the service agreements (as defined in the target reference architecture). In particular, the definition of a conceptual schema/ontology for the description of the service is important in order to address the issue of differences of (i) meanings, (ii) descriptions and formats of the data, and (iii) the concepts held by different cooperating subjects that now should be shared and accepted by all the subjects involved.

According to an "incremental approach," for possible cases in which PAs may not be ready to use rigorous/formal approaches, as a simplified tool, we propose the description of the characteristics of the services using a service/description form whose rows represent the services, while the columns represent the service characteristics. Each service should be defined through

- a description;
- a set of inputs;
- the set of outputs; and
- non-functional requirements (e.g., quality of service, security).

The activity consists in the description of the operations enacted by each service (i.e., its interface) and in the representation of the conversations among the services involved in the process through a suitable modeling tool (e.g., UML sequence diagrams). In the description of conversations in an eGovernment scenario it is important to know the point of the process in which an operation can be invoked. This is different from the description of the internal process of a single service, i.e., the description of the workflow implemented by the service to offer such operations.

3. *Realization*[1] *of the operations enacted by each PA involved*: The aim of this activity is the realization of the operations that must be enacted by each PA involved in the macro-process. The inputs are the previously obtained diagrams and the assessment of the technologies in use (e.g., legacy systems). After the formal definition of the macro-process with the identification and the description of (i) the data exchanged, (ii) the interfaces/operations enacted, and (iii) the conversations among the component services, in this activity the following tasks are carried out:

   a. The identification of the databases and of the applications involved in the macro-process that must be wrapped in order to be exported and used. The chosen databases contain the data owned by the PA and exchanged in the macro-process, while the chosen applications realize the operations that will be offered by the services (such databases should have been identified during the state reconstruction step, see Chap. 5).
   b. Choice of the middleware to export the applications: Indeed, each service can be internally implemented in different languages and technologies (coherent with the internal technologies of each organization), and externally it is exposed through a Web service interface (i.e., WSDL).
   c. Design of wrappers, that is of the software level that hides the actual implementation of the functionalities of the system of a PA and presents them through a well-defined service interface.
   d. Realization and deployment, if not already in place, of the domain gateway, which is basically the container of the services exposed by the PAs.

   In summary, this activity leads to the definition of service agreements (cf. eGCSS of Chap. 10) for all the PAs involved in the process/service.
4. *Realization of the macro-process*: This is the actual realization of the composite service provided by the set of PAs involved in the macro-process. Given the descriptions of the interfaces of the component services and of the data exchanged and given the description of the choreography/orchestration representing the conversations among services, in this activity the macro-process is technically specified and deployed on the composition/orchestration engine of the service owner in order to manage the communications among component services and to orchestrate the component services, thus coordinating the operations. If not already in place, the cooperation gateway is also realized. Moreover, now the cooperation agreement is formalized and realized.
5. *Revision of the pre-existing conceptual schemas/ontologies*: The objective of this activity is the realization of a revision process of the ontologies/conceptual schemas describing the pre-existing services, in order to update the repository

[1] With the word "realization" we mean a very precise and concrete definition of all the elements. Indeed we are in an operational planning phase, and no software realization is performed here. All the artifacts outcome of this and the following steps are useful design documents to be used as technical documentation for possible tenders, etc.

of services and descriptions/agreements. Indeed, after an initial period in which the first services start to be deployed, composed, and orchestrated in order to obtain macro-processes, the cooperative system will be populated by a set of simple and composed services. Each service will be characterized by its own semantics description and by its own interface. Therefore, after a while it will be necessary to revise the service descriptions in order to adjust them to conditions that possibly are changed in the meanwhile. Concretely, in this activity the tables, diagrams, and XML files representing the services are revised and updated.

11.2 Tools for the Design Time

In this section we want to briefly recapitulate the modeling tools that the system/software designer can use in order to realize the design steps of our guidelines; in particular, in the previous section we have identified three main modeling tools:

- Business process modeling notation (BPMN): It is used to represent the workflow logic of the process(es) and to model parallel processes and their synchronization. In our guidelines BPMN diagrams are used for the design of the flow representing the macro-process. Appendix B presents a very brief introduction to BPMN.
- UML sequence diagram (SD): It is introduced to represent the dynamics of the communications between the main actor (the client) requiring the service and the PAs. In our guidelines the SD is also used to describe the conceptual model of conversations among the services composing the macro-process. An alternative, more similar to the possible concrete XML syntax for describing conversations, is to use UML state diagrams.
- UML class diagram (CD): It is an alternative to the ER model used to represent the conceptual schema/ontology needed for the description of the component services. UML class diagrams are very similar to ER diagrams. The advantage of using UML class diagrams instead of ER model lies in their inclusion in the same suite of models of sequence diagrams, namely UML [159].

Clearly a lot of tools, either commercial or open source, are available in order to support the system/software designer in the production of such design artifacts.

11.3 Dealing with Legacy Systems

11.3.1 Legacy Systems Classification

Applications and legacy systems (LSs) (see Fig. 11.2) can be classified as follows:

1. *Highly decomposable*, when they are well structured and present some fundamental properties: application modules are structured in three logic layers: (i) a presentation logic, an application logic, and an access/presentation logic; (ii) application components are independent (there are no hierarchical

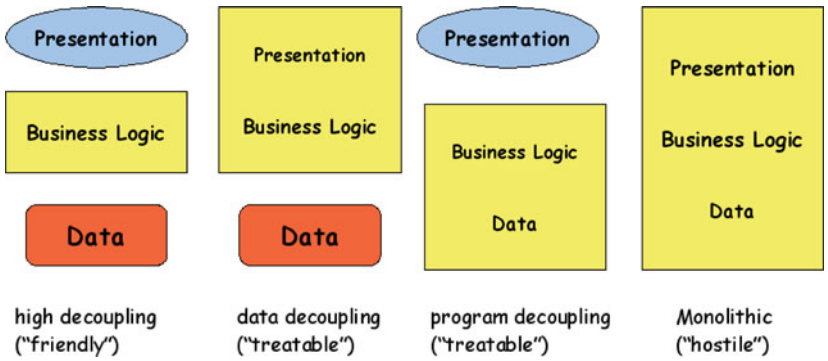

Fig. 11.2 Legacy systems classification

interconnections); (iii) application modules have well-defined interfaces with the database, the access/presentation logic, and the operations exported from other applications.

2. *Data decomposable*, when they are semistructured systems based on the following characteristics: application modules are separated into two layers, namely the modules for data access and the modules for presentation and application logic (the latter ones included in a unique layer). In these systems it is possible to directly access to data, but it is not possible to use the application logic without the presentation/access one.
3. *Program decomposable*, when they are also semistructured with the following properties: modules can be divisible into two layers, namely presentation and data access and application logic (the latter ones included in the same layer). In these systems data are not directly accessible, but it is necessary to invoke predefined functionalities/procedures for accessing them (e.g., stored procedures, CICS transactions). Most legacy applications are classified in this category.
4. *Unstructured*, when they are systems where a single module realizes all three layers. These systems are usually accessible through a terminal.

From the point of view of difficulties in dealing with them, legacy systems can be classified into three categories:

- *Hostile*: Legacy systems classified in this category are not easily interfaceable with external components.
- *Manageable*: Communications with these systems are possible only through ad hoc technologies and developing dedicated interfaces.
- *Friendly*: Interfaces are well defined and easily accessible.

Clearly, highly decomposable systems are friendly, data/program decomposable systems are usable, while unstructured ones are hostile.

11.3.2 Management of Legacy Systems

A system is subjected to a long list of activities for its change and evolution: *assessment*, *maintenance*, and *management*.

Concerning the assessment, here we use the term with a wider meaning than in Chap. 7. In this context the strategy and objectives of organization are evaluated; moreover, the system is analyzed to define if it is valid and usable in order to satisfy organization objectives and consequently to decide to modify the system. Cost analysis has a fundamental role in this phase: the assessment and consequent modifications can be executed only if their costs are not comparable to the cost needed to develop a new system.

Moving to maintenance, for a long time it was the only activity associated with the evolution of a system. In fact, systems were developed, deployed, and only updated for correcting errors, until their substitution. Recently, management is also acquiring a fundamental relevance, but it can require a deep knowledge of system that we want to modify (white box) or can require only knowledge of external interfaces of our system (black box) [213]. The *white-box* approach requires reverse engineering emphasizing a deep knowledge of different modules composing the entire system (program understanding) and activities of reconversion. The *black-box* approach studies only external interfaces of the system and it defines the encapsulation needed to work with them (wrapping). Reverse engineering is a process of analysis whose principal aim is to identify components of a system and their interactions in order to create a representation at an higher abstraction level. Starting from this representation, it is possible to substitute the system (through a clean substitution or gradually), reconverting its code to a new environment, to develop ex novo some parts or adding some new components. It is a slow, specialistic, and expensive process that has high risks and requires miscellaneous knowledge.

Wrapping is a different approach that tries to avoid needing to know the internal structure of the system. It builds a software module around some units that have well-defined interfaces. Wrapping, coherently with the object-oriented paradigm, is based on encapsulation and information hiding and requires less work than reverse engineering. Useful approaches to manage a legacy system are the following:

- *Ignore*: the system is omitted from the next development.
- *Clean substitution*: the system is developed from scratch.
- *Gradual migration*: the system is gradually transformed.
- *Integration*: the system is integrated in future applications without modifications and using wrapping with ad hoc technologies.

Among these approaches we can distinguish two final objectives: substitution of a system with a new one, or its maintenance, adapting to a new environment (integration).

For a long time, the only possibility was substitution, implemented through a single step (clean substitution) or incremental steps (gradual migration): in any case the aim was to create a new system which could reproduce and extend functionalities and data of the old system. Recently, an evolution of the substitution approach

has emerged, based on the idea of leveraging legacy, resulting in the integration of legacy system, based on middleware and the service technologies and the concept of wrapping. In the literature, there is no systematic nomenclature of all the approaches. Consequently, authors can use different terminologies, for example, *integration* can be used to define integration problems in a broader context. Here we use the term *management* when we want to refer to the general problems, *migration* to specify an incremental substitution, and *integration* to indicate the presented approach where we do not want to completely substitute the system but reuse some of its components, suitably wrapped. Other different approaches can often be used. For example, *revamping*, when the system is cleaned in its external component and then in its user interface. More generally, revamping does not change the systems core but it restricts its activities to create a better interface.

There are three possible strategies to deal with such systems:

- *Application engineering/system replacement* consists in the development of a new information system from scratch.
- *System migration* consists in a gradual modification of the system.
- *Application re-engineering* [100] consists in the reuse of some components of the legacy systems (i.e., reuse of working elements).

In eG4M this last approach is proposed and implemented through the study of external system interfaces and their wrapping, avoiding a deep comprehension of the system technology, building a software case (wrapper) containing the units of the systems with well-defined interfaces.

A *wrapper* is a software level hiding the actual implementation of the functionalities of the system while presenting them through a well-defined object-oriented interface; thus the old-wrapped application appears as an ordinary object of the external world. In the use of wrappers there are two most common behaviors: the first consists in the attempt to use the old system as much as possible with the wrapper which has the aims to export the application interface in the new environment. This way no new functionalities are added, while old applications are made available in the new system. A more mature solution (that can be defined *object-oriented strategy*) brings a real integration for the system; in such an approach the new application domain is built and afterward it determines the set of functionalities of the legacy application it needs. Frequently, what happens is that a small percentage of the functionalities of the legacy system are really useful in the new context. With such a solution the component specification is driven by the necessities of the organization more than by the constraints belonging to the legacy [143, 145].

11.4 A Case Study

In this section the above guidelines are applied to the two sets of services considered in Chap. 3, namely the change of residency bundle and the medical examination request. We have seen that according to the “old” mechanism, the citizen has to

spend a lot of time acting as a "messenger" among the several PAs involved. The aim of redesign is reducing the time spent for its provision while increasing the efficiency.

According to what we have previously discussed, the design phase is started with (i) the identification of the external service (in our case the change of residency), (ii) the identification of the involved PAs, and (iii) the evaluation of the interactions among them and with the citizen requiring the service. We assume that the service for the change of residency is provided in a given territory through physical desks. In order to obtain the service, citizens have to go physically to the desks during their opening time and must fulfill a set of other requirements, e.g., they need to produce certificates or documents and fill in forms. Each information to be acquired, provided, and modified usually needs several interactions with the front offices of the administrations, together with several interactions with the back offices of the involved administrations.

The component services of the macro-process we intend to automate are as follows:

- S1: request of the old certificate. The citizen notifies the change to the municipality he/she is leaving. The involved PA is the municipality A (MA).
- S2: request of the new certificate. The citizen notifies to the local administration where he/she transfers the new address. The involved PA is the municipality B (MB).
- S3: communication of the new address to the police department (PD).
- S4: once obtained the change of residency, update the local registry of health-care beneficiaries. The involved PA is the local health authority (LHA).
- S5: medical exams reservation. The involved PA is the LHA.
- S6: if the citizen has a driving license, update the residence information in the driving license. The involved PA is the traffic control authority (TCA).

In a non-cooperative scenario, we can assume that the involved PAs communicate indirectly through the mediation of the citizen that has the burden of the exchange of necessary information and documents. We can also assume that PAs communicate directly only to verify the correctness of the procedure, e.g., when the PD receives the communication of new address, it verifies that the citizen is registered to the municipality B, asking for information to MBB directly.

The *analysis of the macro-process* is a refinement of the results of the analysis carried out in the state reconstruction step (see Chap. 5). At this stage it creates the PA/services matrix containing

- the PAs involved in the process;
- the simple services that will compose the macro-process;
- the input information and data for the service;
- the output of each service.

With reference to the previous example, we can produce Table 11.1. On the basis of evaluation of the output and input of the several services, it is possible to find out a sequence of building services for the definition of the macro-process. In particular

Table 11.1 Set of services of PAs involved in the macro-process

PA	Service	Input	Output
Municipality A	S1: old certificate request	i.1 citizen's birth i.2 citizen's name i.3 citizen's address	o1. old certificate o2. CCA db updated
Municipality B	S2: new certificate request	i.1 citizen's birth i.2 citizen's name i.4 citizen's old certificate i.5 citizen's new address	o3. new certificate o4. CCB db updated
Police department	S3: communication of the new address	i.6 new certificate	o5. PD db updated
LHA	S4: enrolment in the LHA registry S5: medical exams reservation	i.6 new certificate i.7 medical prescription	o.6 LHA db updated o.7 medical booklet updated o.8 exams done
TCA	S6: update of the information on the driving license	i.6 new certificate i.8 driving license i.9 new medical booklet	o.9 TCA db updated o.10 driving license updated

we represent the internal workflow of the process through a BPMN diagram and the specification of the dynamic of the system through a UML sequence diagram representing the interactions among the involved subjects showing the temporal sequence of the exchanged messages. The two diagrams are shown in Figs. 11.3 and 11.4.

In the next step of *hypothesis for the automation*, the old process is elaborated on the basis of the previously defined models. In this step, some basic aspects of the component services and of the complex process are defined. In particular we need to provide

- a description of the component services;
- the description of the operations they enact;
- the conversations (i.e., the communication paths among services);
- the non-functional properties.

According to our incremental approach, we start with a service/description matrix for the description of the services and afterward we move to UML diagrams. A possible example of matrix for the systematic description of the component services is shown in Table 11.2. In the matrix each service is defined through (i) a description (plain text in the example), (ii) the required input, (iii) the produced output, and (iv) a description of non-functional properties.

A more advanced technique would consist in the use of an extended UML class diagram representing the conceptual schema of the services involved. Such a diagram is extended as it represents both the concepts used as input/output of the services, the administrations, and the input/output relationships among them. The

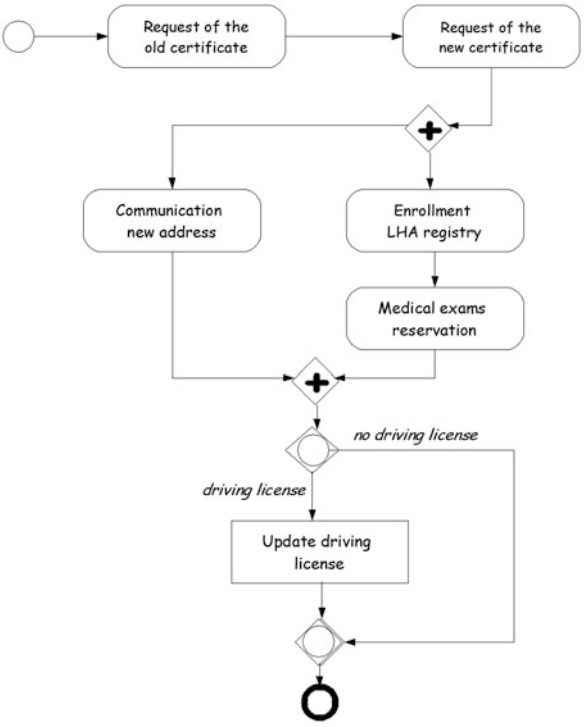

Fig. 11.3 The diagram representing the workflow of the process

reader interested in specific techniques for the use of class diagrams for representing information flows among services can refer to [47].

At this point the design of conversations of the services involved in the *new process*, that is the representation of dynamic of the system in the new automatic approach, follows. Such a representation is realized with a UML sequence diagram representing the communication path in the macro-process and the temporal sequence of the actions enacted in order to realize such macro-process. The diagram is shown in Fig. 11.5. Here we suppose to use Web services, each one deployed on the domain gateway of the owner PA. In the figure, Web service of a PA is indicated as WS@PA (i.e., the Web service of municipality A is indicated as WS@MA).

In the previous steps we have described the composite services and analyzed the macro-process we want to automatize; with these elements the fourth phase of methodology consists in the definition of *operations enacted by each PA*. After the choice of suitable middleware technology (the WS technology) supporting communications among PAs, this step consists in the building of wrappers for the management of different technologies that implement the services for each PA represented in the example. In this step the operations belonging to the simple component services of each PA are wrapped and exposed to the other PAs in the domain gateway.

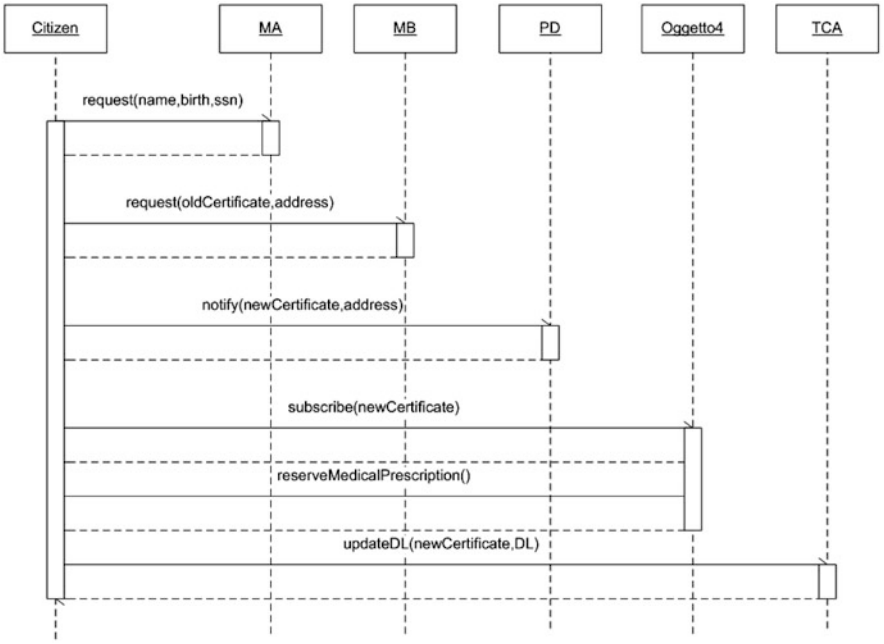

Fig. 11.4 The sequence diagram representing the dynamic of the old process.

Table 11.2 Systematic description of the services

Service	Description	Input	Output	Non-functional req
S1. old certificate request	The service provides a certificate to attest the residence of the citizen. Given the following input it returns the specific output	Citizen's birth Citizen's name Citizen's address	Old certificate MA db updated	nf1. the data of the citizen should be sent via Internet through a secure connection
S2. new certificate request	The service provides a certificate to attest the new residence of the citizen	Citizen's old certificate Citizen's name Citizen's birth Citizens's new address	New certificate MB db updated	nf1. nf2. the data of the different db should be coherent
S3. communication of the new address	The citizen communicates to PD the new address for possible controls	New certificate	PD db updated	nf2.

Table 2 (Continued)

S4. enrollment in the LHA registry	The citizen is enrolled in the medical registry	New certificate Medical – prescription	LHA db updated medical booklet updated	nf2. nf3. enrollment done assuring the privacy
S5. medical exams reservation	Before the enrollment the citizen needs to make exams	Medical – prescription	Exams done	nf3.
S6. update of the information on the driving license	With exams and the new address the citizen has to update his/her DL	New certificate Driving license New medical - booklet	TCA db updated Driving license updated	

The last step is to define the orchestration identifying the macro-process in a suitable language (i.e., WS-BPEL) and such a description is deployed on a suitable orchestration/composition engine.

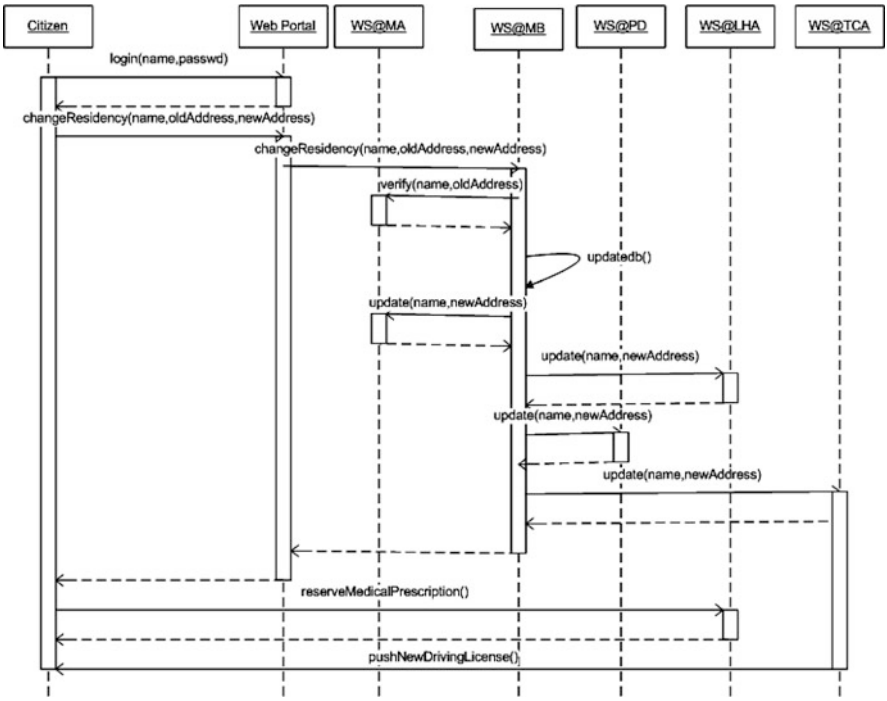

Fig. 11.5 The sequence diagram representing the dynamic of the system after the process of automatization

11.5 Summary

In this chapter we have provided guidelines for the specification of (external) services requiring the automation of macro-processes. Such guidelines are the last activity of the operational planning. The artifacts produced are used as technical specifications for the effective software project, both in the case if a project is internally realized by the PAs involved and in the more common case if a tender is launched for such an activity.