

EA4UP: an Enterprise Architecture-Assisted Telecom Service Development Method

Jacques Simonin¹, Francis Alizon¹, Jean-Pierre Deschrevel²,
Yves Le Traon³, Jean-Marc Jézéquel⁴ and Bertrand Nicolas¹

¹ Orange Labs, 2 avenue P. Marzin 22300 Lannion, France,

² Orange Labs, 42 rue des Coutures, 14066 Caen, France,

³ ENST Bretagne, 2 rue de la Châtaigneraie, 35576 Cesson-Sévigné Cedex, France,

⁴ INRIA & Rennes University, Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France

jacques.simonin@orange-ftgroup.com, francis.alizon@orange-ftgroup.com,
jeanpierre.deschrevel@orange-ftgroup.com, yves.letraon@enst-bretagne.fr, jezequel@irisa.fr,
bertrand.nicolas@orange-ftgroup.com

Abstract

The cost of a telecom service development is correlated to the discontinuity and the complexity of the process. To solve this problem, we propose a method dedicated to telecom service development, called EA4UP method. The first EA4UP characteristic is the use of the Enterprise Architecture (EA) for the design activity. The EA promotes component reusing, and improves development process continuity thanks to a Model Driven Engineering approach. In this new method, EA enforces the transformation of an analysis model into a design model. The second EA4UP characteristic is to place functions in the core of the method (instead of data). The assessments of eleven projects allow to measure profits of this EA4UP method with regard to the previous ones.

Keywords: Unified Process, Enterprise Architecture, telecom service development.

1. Introduction

A telecom service is a service provided to an end-user by a telecommunication operator. For example, a messaging service enables an end-user to create an email, to send it to another user and to receive emails. This service is decomposed in three functions: message creation, message sending and message receipt. In France Telecom, architects develop new telecom services based on such decompositions which are instantiated into components. For decreasing

development costs, components reuse becomes the rule and not the exception. Such reuse requires identifying and sharing service components. Besides, telecom service development processes follow the standards of system development [1], for instance they use a development process such as the Unified Process (UP) [2], well-known incremental, iterative use-case driven process.

One of the major requirements of telecom service development is that they should leverage existing building blocks called enablers, as defined by Open Mobile Alliance (OMA) [3]. An enabler dedicated to messaging functions could be used for the architecture design of the previous messaging service. It would indeed make no sense to redevelop an enabler implementing basic messaging functions with a specific protocol as POP [14]. More generally, the current Information System (IS) infrastructure must be optimized to allow the smooth integration of new telecom services. This reusing means also a telecom service architecture rationalization with a sharing of functions and technologies. The telecom services IS evolution should then benefit of this rationalization.

A dynamic approach is intrinsic to service design. Consider for example a simple electronic message sending service that can be described as a sequence of: (i) a successful user authentication; and (ii) a message sending by the user. For business IS applications, the static aspect description is often the main point of view with the study of the data model. The Orange Labs experience is that the data driven approach fails in capturing the telecom service dynamics. With a data model, the architect has to design data from entities

specified during the requirements workflow [9]. For example, a messaging service scenario could be:

1. *a message is sent by a user identified by name*
2. *the message is stored in the messaging box of the user*

Data could be *User's identity*, *Message* and *Messaging box*. Then, the architect must design dependencies between these data according to the telecom service scenario. The scenario becomes:

- a dependency between data extracted from *Message* and *User's identity*
- a dependency between data extracted from *Message* and *Messaging box*

A telecom service is less complex to understand with a function sequence description than with a data model description. For example, a messaging service scenario could be:

1. *Identify the messaging user*
2. *Send a message*
3. *Store a message in a messaging box*

This paper presents a new method, called EA4UP (Enterprise Architecture for Unified Process), which is a customization of UP to deal explicitly with dynamic aspects of telecom service architecture. Used by France Telecom, its specificity lies in the design step which is primarily seen as a composition of enablers. The enabler definition and production are the goals of the Enterprise Architecture (EA) [4] of the telecom services Information System (IS).

To be able to reuse enablers, the principle of our solution is based on the dynamic aspect of a telecom service. This re-use objective requires the precise study of all steps of every telecom service scenario, and in particular, to cater for all messages exchanged between enablers.

Section 2 presents how the EA is used in a UP development of a telecom service. This section focuses on the description of the EA4UP method for functional and technical designs. Requirements and analysis workflows are consistent with the previous France Telecom approach [5], and design, implementation and test workflows are not specific in our approach. Section 3 presents related works. The EA4UP method has been evaluated in the field at both France Télécom and Telekomunikacja Polska, and experimental results are reported in Section 4, as well as the perspectives deduced from these results.

2. EA4UP method

The main characteristic of the EA4UP method is to involve the functional telecom services EA. The functional architecture defines the generic functional

elements for telecom services. These elements can be specialized according to each telecom service. In the same way, the technical EA defines a set of recommendations of technical elements which are used in the design phase; for example, the protocols on which the service will be deployed.

The outcome of each activity in the workflow is captured in an abstract way through a model, expressed using the Unified Modeling Language (UML) [7].

The EA4UP method for telecom service design is an iterative sequence of tasks. Each task is characterized by its input and output elements. Every input element is the result of an upstream task. This paper details only the functional and technical designs enforced by, respectively, functional and technical EA.

The messaging telecom service outlined in the introduction will serve as a running example to illustrate our EA4UP method.

2.1 Method overview

The EA is the solution at the core of the EA4UP method to solve the development cost problem. The functional and technical EAs provide the functional and technical frameworks useful for component sharing. The EA allows the re-use of the enablers, hence making it possible to decrease the development costs.

The MDE approach [6] where an analysis model is transformed into a functional design model or into a technical design model will be used to address the problem of development discontinuity. Telecom service architects have to enforce the EA of telecom services IS. The EA playing the role of transformation enforcing, enables thus the improvement of the development continuity.

To decrease the telecom service development complexity, data are deduced from function interactions which illustrate the telecom service usage. The data design is thus a consequence of the design of the dynamic aspects of the functional architecture, and it is not an independent design.

The 5 core workflows defined in an iterative UP development process are: Requirements, Analysis, Design, Implementation and Test [1]. Within the framework of a service telecom development described in Fig.1, the core Design workflow is split along architectural views [12]:

- Functional Design for the functional view which contains the IS functions carrying out the telecom service analysis ;
- Technical Design for the technical view which describes the set of technologies used for the telecom service deployment ;

- Applicative Design for the applicative view which describes the set of telecom service applicative elements.

The functional EA impact takes place between Analysis and Functional Design. Functional EA is designed by enterprise architects who are experts of the telecom services IS areas. They design a functional view which corresponds to the strategy of the company. In the case of France Télécom, the telecom services IS functional view is composed of several tens of functional components.

The architects who design the target functional view of the telecom services IS also add functional interfaces to these functional components. These interfaces must be compliant with the strategy of the company. For example, the enterprise architects may consider that, strategically, a mailbox has to refer to the user's identity of a messaging service. In that case, in the target functional view, they would design a functional interface provided by the identity management functional component and used by the messaging box management functional component.

The impact of the technical EA also takes place between Analysis and Technical Design. Technical EA provides to telecom service architects a target technical view of their IS. It is designed by enterprise architects who are experts of technologies used in the IS areas of telecom services. They design a technical view which corresponds to the strategy of the company. For example, the technical strategy may target

- a telecom service enabler, e.g. a messaging enabler,
- a network protocol, e.g. SIP (Session Initiation Protocol).

The technical architect of a service telecom has to respect these recommendations. In the case of telecom service, Applicative Design is concerned because the applicative view of an enabler is deployed on its technical view.

2.2 Functional Design

The Functional Design workflow which comes from the Design workflow is central in the EA4UP method. It is the workflow where the telecom service architect takes into account the target functional architecture of the IS. Every task concerned by the functional EA is qualified as generic. When a task is only concerned by the current telecom service, it becomes specific to this service.

The Functional Design workflow is split into 3 tasks:

- Static and generic functional design of functional components ;
- Dynamic and generic functional design of functional interfaces ;
- Specific functional design of output parameters of functional interfaces.

2.2.1 Static and generic functional design

The rule associated with this task is represented in Figure 1: each telecom service functional component is deduced from at least one entity or one link between entities specified during the "Static functional analysis" task and is constrained by functional components designed in the functional EA. Functional components are captured in a UML component diagram.

The functional components are selected in the functional EA of telecom services IS described in the Introduction. The goal of this task is to select among the functional components of the functional EA, those participating in the implementation of the use described in the Requirements workflow. To make this selection easier, the functional architects of company detail each component by its functions.

It may happen that a functional component needed for the implementation of the use case does not yet exist in the functional EA. A negotiation is then recommended between the architects of the telecom service and the enterprise functional architects of company to validate, or not, the creation of this new component in the target functional view of the IS. This recommendation is used for the enhancement of the functional EA.

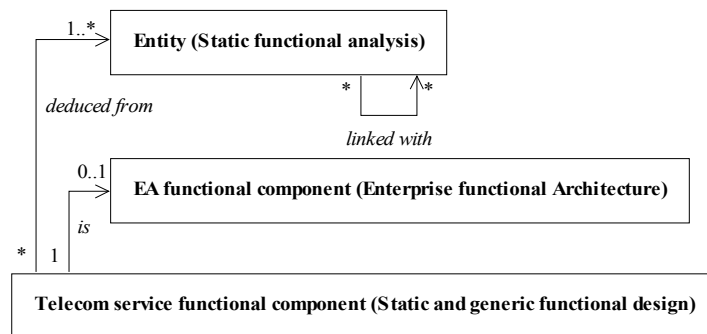


Figure 1. Static and generic functional design task

In the illustration, the entities specified during the "Static functional analysis" task are represented in Figure 2.

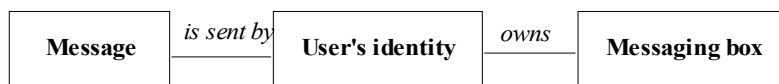


Figure 2. Analysis model (entity) illustration

The functional components designed during the "Static and generic functional design" task deduced from this entity model and extracted from the functional EA are represented in Fig. 3:

- the function of message sending defines the functional component *Message Sending*,
- the function of user identification defines the functional component *Identity Management*,
- the function of message storage defines the functional component *Message Storage*.

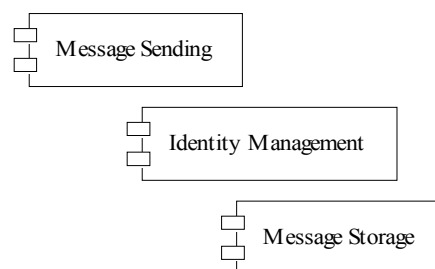


Figure 3. Functional design model (component) illustration

2.2.2 Dynamic and generic functional design

The rule associated with this task is represented in Figure 4: each functional interface designed during the

"Static and generic functional design task" (see Section 2.2.1) carries out at least an interaction between entity instances specified during the "Dynamic functional analysis" task and is constrained by functional interfaces designed in the functional EA.

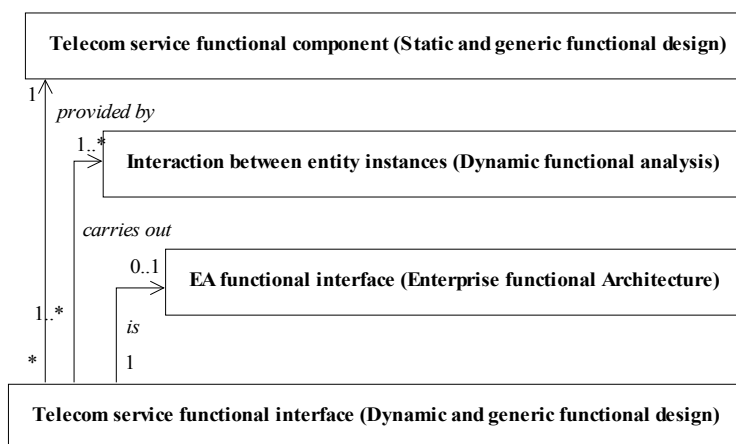


Figure 4. Dynamic and generic functional design task

During this task, each step of the scenario is transformed into a sequence of interfaces provided by functional components selected in Section 2.2.1. As for functional components, it is possible that a required functional interface does not yet exist in the functional EA. A negotiation is also recommended between the architects of the service telecom and the enterprise

functional architects of company to validate, or not, the creation of this new interface in the target functional view of the IS.

The interactions between entity instances specified during the "Dynamic functional analysis" task are represented in Figure 5.

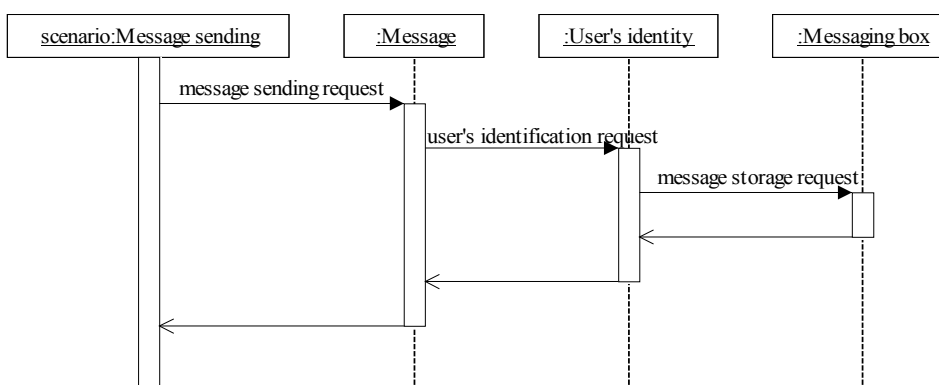


Figure 5. Analysis model (entity instances interaction) illustration

The functional interfaces designed during the "Dynamic and generic functional design" task are deduced from this model of the interactions between entity instances and extracted from the functional EA as represented in Figure 6.

The link between provided functional interfaces and the scenario is the following one:

- the functional interface *sendMessage* provided by the component *Message Sending* carries out the interaction *message sending request*,
- the functional interface *getIdentity* provided by the component *Identity Management* carries out the interaction *user's identification request*,
- the functional interface *storeMessage* provided by the component *Message Storage* carries out the interaction *message store request*.

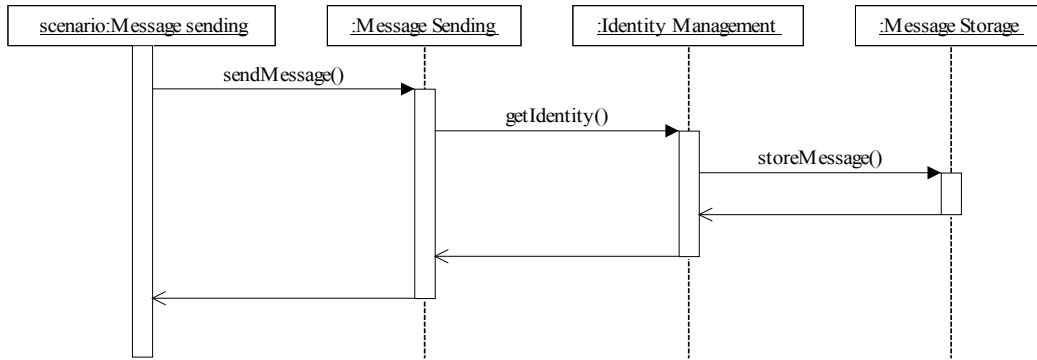


Figure 6. Functional design model (components interaction) illustration

The chain of interfaces is deduced from their dependency in the scenario. The resulting functional component diagram is illustrated in Figure 7.

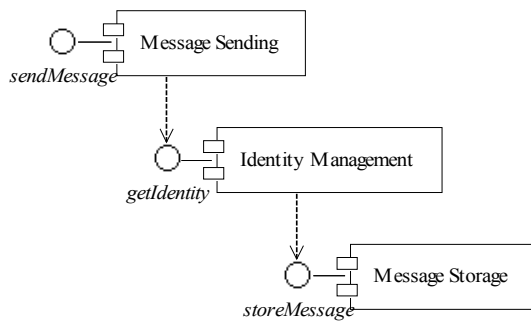


Figure 7. Functional design model (interface) illustration

2.2.3 Specific functional design

The rule associated with this task is represented in Figure 8: each functional interface output parameter is extracted from at least one entity specified during the "Static functional analysis" task and is constrained by a functional interface designed during the "Dynamic and generic functional design" task (see Section 2.2.2). A UML sequence diagram captures each interaction between functional component instances of a scenario defined during the Requirements workflow and its output parameters resulting from the interaction.

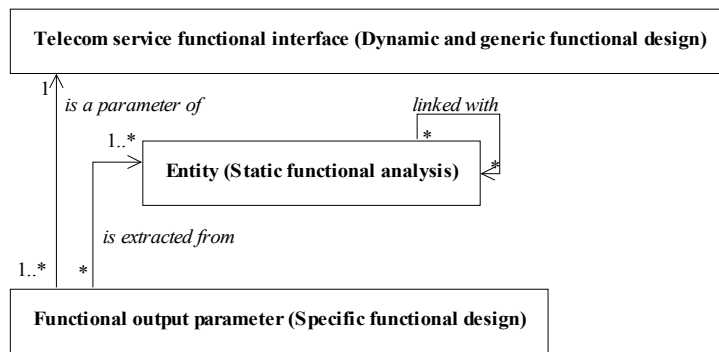


Figure 8. Specific functional design task

The telecom service specificity of the functional design task is captured by the following rule: each output parameter is extracted from an entity participating to a telecom service use case. An output

parameter of a functional interface is obviously produced by the functional component providing the interface.

Figure 9 represents output parameters of each functional interface required to realize the message sending scenario:

- the output parameter *Message - Message Sending* of the functional interface *sendMessage* is extracted from the entity *Message* and produced by the component *Message Sending*,
- the output parameter *User's identity - Identity Management* of the functional interface *getIdentity*

is extracted from the entity *User's identity* and produced by the component *Identity Management*,

- the output parameter *Messaging box - Message Storage* of the functional interface *storeMessage* is extracted from the entity *Messaging box* and produced by the component *Message Storage*.

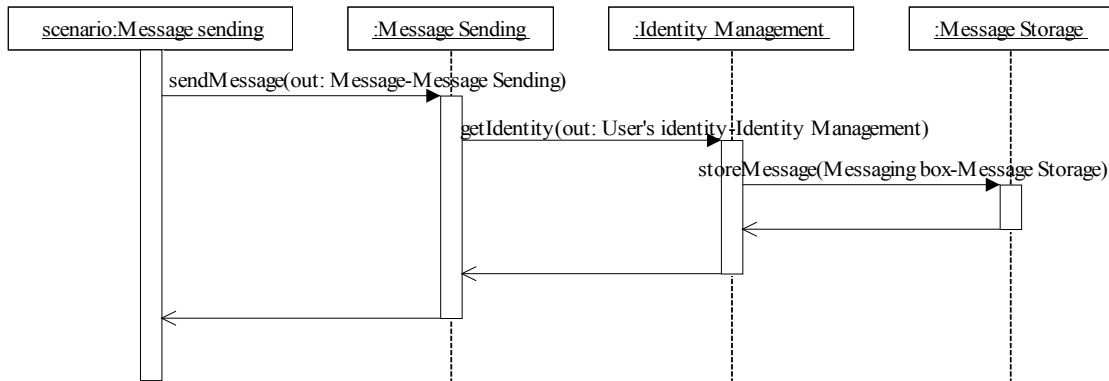


Figure 9. Functional design model (output parameter) illustration

Figure 10 represents the data model deduced from the designed output parameters. Each interaction involves a dependency between output parameters seen

as data. Each dependency is then drawn up with the functional interface characterizing the interaction.

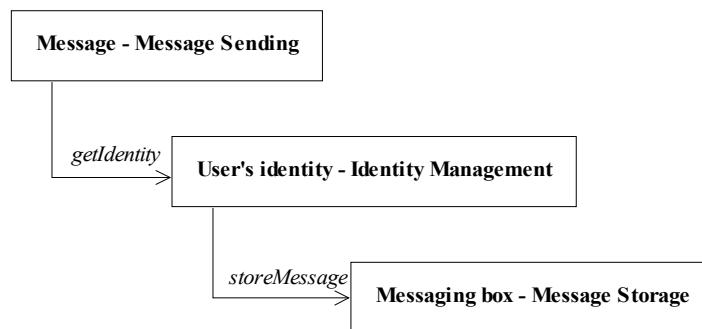


Figure 10. Functional design model (data) illustration

2.3 Technical Design

Technical Design workflow is extracted in a similar way from the Design workflow. Nodes and protocols connecting nodes are the only concepts of the Technical Design workflow, which is split into 3 tasks:

- Static technical design of the nodes of the deployment infrastructure

- Dynamic technical design of the protocols of the deployment infrastructure
- Detailed technical design of nodes and protocols

2.3.1 Static technical design

The rule associated with this task is represented in Figure 11: each node fulfils a static technical property

specified during the "Static technical analysis" task and is constrained by a technical EA recommendation, especially for the technical choice of enabler. The telecom service architect must check this choice with non-functional requirements of the developed telecom service. A UML deployment diagram can be used to capture this.

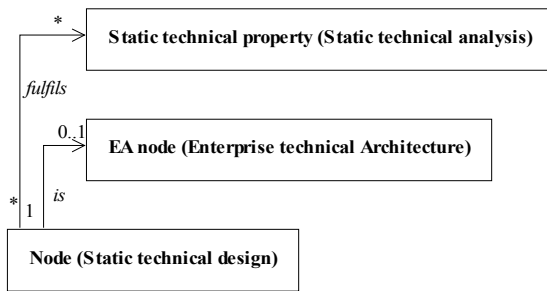


Figure 11. Static technical design task

In our example, during the "Static technical analysis" task, the specified static technical property is the link between the non-functional requirement of *Storage maximum capacity of the messaging box per user* and the *Messaging box* entity. The choice of the *Storage enabler* during the "Static technical design" task represented in Fig. 12 fulfils this requirement and is consistent with technical EA recommendations:

- the node *Storage enabler* fulfils static technical property associated with the entity *Messaging box*,
- the node *Messaging enabler* is associated with the entity *Message*,
- *Messaging enabler* and *Storage enabler* may be every one of them associated with the entity *User's identity*.

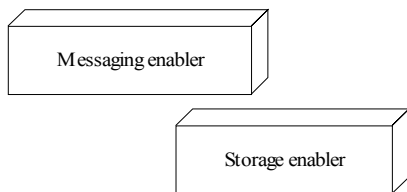


Figure 12. Technical design model (node) illustration

2.3.2 Dynamic technical design

The rule associated with this task is represented in Figure 13: each protocol connecting nodes (see Section 2.3.1) is deduced from at least one dynamic technical

property specified during the "Dynamic technical analysis" task and is constrained by a technical EA recommendation. The telecom service architect must choose protocols according to non-functional requirements of the developed telecom service. This is captured in a UML deployment diagram.

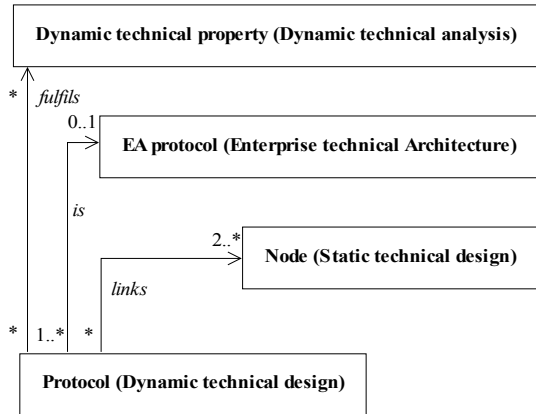


Figure 13. Dynamic technical design task

In our example, the *FTP* protocol fulfils the dynamic technical properties specified during the "Dynamic technical analysis" task:

- the link between the interaction *user's identification request* (from *Message* to *User's Identity*) and the non-functional requirement for *an average user are 10 sent messages per day*,
- the link between the interaction *message store request* (from *User's Identity* to *Messaging box*) and the non-functional requirement *user messaging box is stored 1 time per day*.

A protocol must then exist between the enablers; whatever may be the enabler associated with the entity *User's Identity*:

- the *FTP* protocol is recommended by technical EA and fulfils the non-functional requirements (see Figure 14).

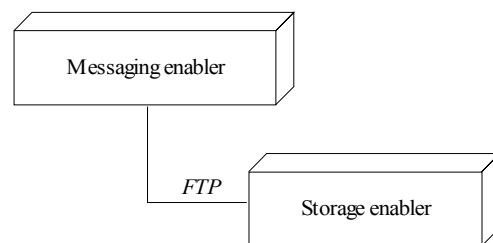


Figure 14. Technical design model (protocol) illustration

2.3.3 Detailed technical design

This task allows detailing of operational environments, physical nodes making up each node (see Section 2.3.1) and each communication link (see Section 2.3.2). This task is not discussed here because there is no specificity about it in this method.

3. Experimental validation

The EA4UP method prototyping ended in June 2007. Eleven new telecom services development projects from France Télécom and from Telekomunikacja Polska have been involved in this field test.

For the analysis framework, on average these projects specified:

- 20 entities
- 50 interactions between entity instances
- 20 technical properties

It is not possible to detail here these numbers by project for a reason of data privacy.

A questionnaire with nine questions has been sent to project architects. Answers to these questions are detailed in Table 1. Four themes emerge in this questionnaire:

- the EA4UP approach itself (Q1: approach very complicated "--", complicated "-", simple "+", very simple "++"; Q2: approach without interest "--", not very useful "-", useful "+", essential "++"),
- the EA4UP approach compared to previous ones (Q3: component reusability definitely worse "--", worse "-", better "+", definitely better: "++"; Q4: quality definitely worse "--", worse "-", better "+", definitely better "++"),
- the EA4UP cost compared to previous approaches (Q5: designing time definitely longer "--", longer "-", shorter "+", definitely shorter "++"),
- the EA4UP communication capabilities (Q6: approach does not help "--", helps a little "-", give sounds background "+", is essential "++"; Q7: to work with marketing team approach is without interest "--", not very useful "-", useful "+", essential "++"; Q8: to work with architects approach is without interest "--", not very useful "-", useful "+", essential "++"; Q9: to work with decision makers approach is without interest "--", not very useful "-", useful "+", essential "++").

Project	Approach		Architecture		Cost	Communication				EA skill
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	
P1	-	++	++	+	+	--		++	++	N
P2	-	++	+		--			++		N
P3	+	++	++	+	+	+	-	+	-	Y
P4	-	+	+	+	-	+	-	+	+	N
P5	-	+				+	-	-	--	N
P6	+	++	+	++	++	++	++	++	++	Y
P7	-	+	++	++	-	++	+	+	+	N
P8	+	+	++	+	-	+		+	+	N
P9	-	+				+		+	+	N
P10	+	++	+	+	++	++	+	++	+	Y
P11	-	++	++	++	-	+	+	++		N

Table 1. Detailed project architect answers to the questionnaire about EA4UP method prototyping from the smallest project P1 to the greatest P11 in terms of number of designed components

The last column representing the EA skill of the architect is added to this results table. This skill is closely connected with favourable answers of the project architect about the method simplicity and the

cost of this approach. The size of the project architecture is on the other hand not explicitly connected with architect answers.

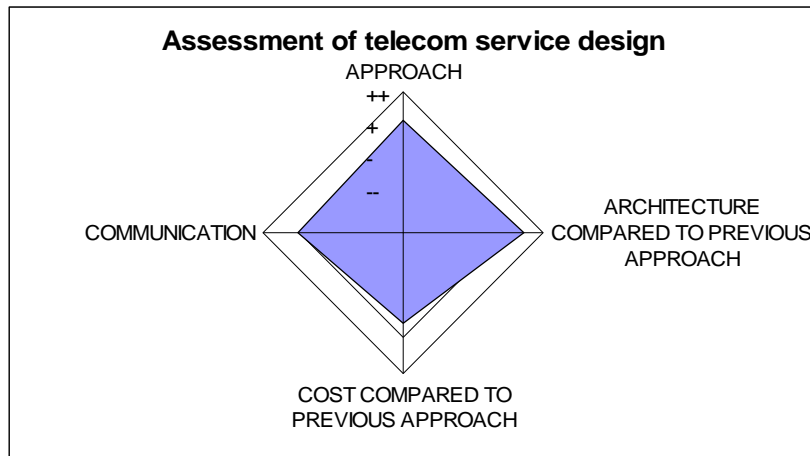


Figure 15. Advantages (+ or ++) and drawbacks (- or --) of the EA4UP method

The best result of the assessment of our method synthesized in Figure 15 is for the comparison between EA4UP method and the previous approaches of the telecom service architects. The result is close to ++ for the component reusability.

The worst result is about the EA4UP cost. The reason is the cost of the functional EA suitability. Project noticing about designing time depends on projects (from definitely shorter to definitely longer). The functional EA knowledge is the main parameter for this noticing. This functional EA consists of several ten of functional components and functional interfaces.

EA4UP approach in spite of its complexity has a good evaluation (+) thanks to its interest for the telecom service architecture design but architects need support. For all projects, this approach is very useful for their architecture design.

With EA4UP method, the communication about the telecom service architecture is easier with other architects (between + and ++). The result is also good (+) for the communication with marketing teams and decision makers, but it is however variable (from - to ++).

4. Related work

Telecom service development is considered as an object oriented development as defined by the Unified Process (UP). UP is moreover adapted for telecom service development because of its iterative feature. In our approach, the design workflow is split into 3 workflows: functional design, technical design and applicative workflows. It makes easier to involve EA constraints during telecom services development.

EA and its framework are suitable to improve telecom service component reuse. EA involved in the development process helps business, functional or technical framework sharing. For example, the SEAM approach [10] makes easier business and IT alignment. This approach is IS cartography centric but does not deal with the whole telecom service development process. Enterprise Unified Process (EUP) [8] enables an integration of UP in EA design. ODP [11] is a well-adapted framework for telecommunications only if view points and concepts are taken into account. Nevertheless it is difficult to use it for development processes because of the lack of link between the technical and information view points.

From a Service Oriented Architecture perspective (SOA) [16], a telecom service is defined as an orchestration of service components. The component orchestration implements the enterprise business processes. The components defined by EA provide an accurate definition of SOA components [15]. Using EA components, the SOA objective of components reuse can be reached easier. However, if the development of SOA components highly benefits from EA, it is not the case with orchestration that does not take advantage from functional or applicative dependencies between components.

The main contribution of this paper is the use of functional and technical EA frameworks during UP development process. To allow a well EA integration, we choose the framework described in [12] with its 4 views: business, functional, technical and applicative. A benefit of this approach is to take into account the links between these views all along the development process. Moreover, the design of the dependencies between functional or applicative components brings

richer component interfaces and introduces interface optimization.

MDA [17] and MDE [6] approaches allow applicative design frameworks integration during development process [16]. EA4UP takes so benefits of the MDA approach. In an unusual way, functional or technical EA define indeed the Platform Definition Model (PDM) which enforces respectively generic functional and technical design tasks.

5. Conclusion

Experimentation results show that the EA4UP method allows an easier reuse of enablers. Thanks to the functional and technical EAs, this reuse implies a telecom service architecture rationalization, with a sharing of functions and technologies.

Telecom services IS evolutions may then benefit from this rationalization. The main need for architects who have been involved in the EA4UP method prototyping is a functional EA skill improvement. Decisions have been taken in Orange Labs to extend this approach.

Future works will try to address the concerns raised on the field experiment mostly the need for support during the design workflow and the high cost for some projects. To address these issues, we are designing and implementing tools for:

- assisting in the functional design workflow according to the functional EA model;
- helping telecom service architects in all their design tasks;
- making the communication about the architecture results easier.

These tools will use MDE techniques, such as model transformations (SmartQVT [13]).

6. References

- [1] International Organization of Standardization, "Information technology -- Process assessment", ISO/IEC PRF TR 15504.
- [2] I. Jacobson, G. Booch, J. Rumbaugh, "The Unified Software Development Process", Addison-Wesley, 1999.
- [3] OMA, "OMA Service Environment", Version 1.0.2, 2005.
- [4] J.A. Zachman, "A Framework for Information Systems Architecture", IBM Systems Journal 26, No. 3, 1987, pp. 276-292.
- [5] B. Nicolas, "MetYs", OMG Information Days, Paris, France, 2002.
- [6] D.S. Frankel, "Model Driven Architecture – Applying MDA to Enterprise Computing", Wiley Publishing Inc, 2003.
- [7] G. Booch, J. Rumbaugh, I. Jacobson, "The Unified Modeling Language – User Guide", Addison – Wesley, 1999.
- [8] S.W. Ambler, J. Nallbone, M.J. Vizdos, "Enterprise Unified Process – Extending the Rational Unified Process", Prentice Hall, PTR, 2005.
- [9] J. Simonin, Y. Le Traon, J.M. Jézéquel, "An Enterprise Architecture Alignment Measure for Telecom Service Development", Proceedings of the 11th IEEE International EDOC Conference, Annapolis, USA, 2007, pp. 476-483.
- [10] A. Wegmann, "On the Systemic Enterprise Architecture Methodology (SEAM)", International Conference on Enterprise Information Systems (ICEIS), Angers, France, 2003.
- [11] OMG, "Reference Model of Open Distributed Processing", ISO/IEC 10746-1 to 4 | ITU-T Recommendation, X.901 to 904, 1996.
- [12] C. Longépé, "The Enterprise Architecture IT project – The Urbanisation paradigm", Kogan Page, 2003.
- [13] SmartQVT, <http://smartqvt.elibel.tm.fr/>.
- [14] IETF, "Post Office Protocol (POP)", RFC 1939, <http://www.ietf.org/rfc/rfc1939.txt>.
- [15] A. Grigoriu, "An Enterprise Architecture Development Framework", Trafford Publishing, 2006.
- [16] R.T. Burlton, "Business Process Management – Profiting from Process", Sams Publishing, 2001.
- [17] OMG, "MDA Guide", Version 1.0.1, 2003.
- [18] N. Guelfi, G. Perrouin, "Using Model Transformation and Architectura Frameworks to Support the Software Development Process: the FIDJI Approach", Midwest Software Engineering Conference, Chicago, USA, 2004, pp. 13-22.