4 Communication of Enterprise Architectures

This chapter presents a communication perspective of enterprise architectures. We provide both a theoretical and a practical perspective of the issues involved in the communication of enterprise architectures. The general idea is that the chapter helps the reader see how architecture development and modelling can be optimally supported by discussing why certain forms of modelling are used in some situation and how this fits the goals in the process. The theoretical perspective will focus on communication during system development in general, where the word system should be interpreted as any open and active system, consisting of both human and computerised actors, that is purposely designed. The practical perspective will take shape as a set of practical guidelines that should aid architects in the selection and definition of architecture description approaches that are apt for a specific (communication) context.

4.1 Introduction

Describing architectures is all about communication. If some architecture description is not used as a means of communication in some shape or form, then this description should not have been created in the first place. Whatever the role of an *architecture description* is, it always involves some communicative aspect. Consider, as an illustration, the potential *uses* of architecture descriptions as identified in the IEEE 1471 standard (IEEE Computer Society 2000):

- Expression of the system and its (potential) evolution.
- Analysis of alternative architectures.
- Business planning for transition from a legacy architecture to a new architecture.
- Communications among organisations involved in the development, production, fielding, operation, and maintenance of a system.
- Communications between acquirers and developers as a part of contract negotiations.

- Criteria for certifying conformance of implementations to the architecture.
- Development and maintenance documentation, including material for reuse repositories and training material.
- Input to subsequent system design and development activities.
- Input to system generation and analysis tools.
- Operational and infrastructure support; configuration management and repair; redesign and maintenance of systems, subsystems, and components.
- Planning and budget support.
- Preparation of acquisition documents (e.g., requests for proposal and statements of work).
- Review, analysis, and evaluation of the system across the life cycle.
- Specification for a group of systems sharing a common set of features, (e.g., product lines).

Each of these uses of architecture involves forms of communication. In this vein, in this chapter we present a 'communication-aware' perspective of enterprise architectures. In doing so, we provide both a theoretical and a practical perspective of the issues involved in the communication of enterprise architectures. The theoretical perspective will focus on communication during system development in general, where the word system should be interpreted as any open and active system, consisting of both human and computerised actors, that is purposely designed. The practical perspective will take shape as a set of practical guidelines that should aid architects in the selection (and definition) of architecture description languages and approaches that are apt for a specific (communication) situation.

Architecture descriptions are used to communicate the architecture of a planned or pre-existing system. This could be a system that is part of an enterprise, an organisation, a business, an information system, a software system, or the hardware infrastructure. The communication about the system and its architecture is likely to take place between different stakeholders of that system.

In this book, the primary focus is on architectural models of a graphical (as opposed to textual or verbal) nature. One may refer to these as architectural models 'in the narrow sense'. In this chapter, however, we are concerned with architecture descriptions in 'the broader sense'. In other words, textual, verbal, or any other types of architecture descriptions are included.

At present, many description languages are already available to architects, while many more are being created by both academia and industry. Why all these languages? How does one select the language that is most

apt in a given situation? Such questions beg for a well-conceived answer. In line with the old adage 'practice what you preach', we argue that just as proper requirements engineering is needed for the development of systems, proper requirements should also be formulated for languages and approaches that are to used as vehicles for communication during system development. In formulating these requirements, several factors should be taken into account, such as the development goals, the communication goals, the concerns, personal goals, abilities, and attitudes of the actors involved, etc.

We set out to provide a theoretical underpinning of the issues involved, as well as practical guidelines that will aid architects in selecting the best approach for their architectural communicative needs. We will therefore start out with a theoretical exploration of the issues involved in communication during system development (Sects. 4.2 and 4.3), followed by the application of this exploration to the field of enterprise architecture (Sect. 4.4).

4.2 System Development as a Knowledge Transformation Process

In essence, we regard system development as a knowledge transformation process whereby conversations are used to share and create knowledge pertaining to both the system being developed, as well as the development process itself. The notion of 'conversation' should be interpreted here in the broadest sense, ranging from a single person producing an (architectural) description, via a one-on-one design or elicitation session, to a workshop with several stakeholders, and even the widespread dissemination of definitive architectures. This way of thinking provides a frame of thought with which one can better understand the (communicative) requirements posed on architecture description languages.

4.2.1 System Development Community

Given our focus on communication, it is important to identify the actors that can play a role in the communication that takes place during the system development process. These actors are likely to have some stake with regards to the system being developed. Examples of such actors are problem owners, prospective actors in the future system (such as the future 'users' of the system), domain experts, sponsors, architects, engineers, business analysts, etc.

These actors, however, are not the only 'objects' playing an important role in system development. Another important class of objects are the many different documents, models, forms, etc., that represent bits and pieces of knowledge pertaining to the system that is being developed. This entire group of objects, and the different roles they can play, is what we shall refer to as a system development community.

> **System development community**: a group of objects, such as actors and representations, which are involved in the development of a system.

(We will clarify below why we regard documents as being part of the community.)

The actors in a system development community will (typically as a consequence of their stakes) have some specific interests with regards to the system being developed. This interest implies a sub-interest with regards to (the contents of) the system descriptions that are communicated within the community. This interest, in line with IEEE 1471 (IEEE Computer Society 2000), is referred to as the *concern* of stakeholders

> **Concern**: an interest of a stakeholder with regards to the architecture description of some system, resulting from the stakeholder's goals, and the present or future role(s) played by the system in relation to these goals.

Some example of concerns are:

- The current situation with regards to the computerised support of a business process.
- The requirements of a specific stakeholder with regards to the desired situation.
- The improvements, which a new system may bring, to a pre-existing situation in relation to the costs of acquiring the system.
- The potential impact of a new system on the activities of a prospective user.
- The potential impact of a new system on the work of the system administrators that are to maintain the new system.

4.2.2 System Development Knowledge

The system development community harbours knowledge about the system being developed. The communication occurring within a system development community essentially is aimed at creating, furthering, and dis-

seminating this knowledge. Depending on their concerns, stakeholders will be interested in different knowledge topics pertaining to the system being developed.

We will now briefly explore the kinds of knowledge that are relevant to a system and its development; in other words, the knowledge topics that can be distinguished. In the next subsections, we will discuss in more detail in what ways this knowledge can be made (more) explicit.

During system development, members of the system development community will create and exchange knowledge pertaining to different topics. We can make a first distinction between the target domain pertaining to the system being developed and the project domain, about the development process itself. We have borrowed these terms from the Information Services Procurement Library (ISPL) (Franckson and Verhoef 1999). For both of these knowledge domains, further refinements can be made with regards to the possible topics. We identify the following additional characterisations:

- **Perspective**: Artefacts, such as systems, can be considered from different perspectives. Some examples are:
  - business, application, and infrastructure aspects of a (computerised) information system;
  - social, symbolical, and physical aspects of a system;
  - process, information, actors, and technology featuring in a system.

  In Chap. 7, the notion of 'viewpoint' will be discussed in depth. A viewpoint takes a specific perspective of a system. The concept of viewpoint is, however, not synonymous with perspective as the former includes some additional items, such as the modelling language that is to be used to describe the system from the given perspective. In contrast, a perspective is purely 'topical'.
- **Scope**: Given a domain, such as a system or a development project, we can identify several scopes when approaching the domain: enterprise-wide, department-specific, task-specific, etc.
- **Design chain**: When considering the design of some artefact, a distinction can be made between:
  - *Purpose*: to what purpose the artefact is needed.
  - *Functionality*: what functionality the artefact should provide to its environment.
  - *Design*: how it should realise this functionality.
  - *Quality*: how well it should do so.
  - *Costs*: at what cost it may do so, and may be constructed.

This distinction applies to the target domain as well as the project domain. In the latter case, the project's execution plan/strategy is the designed artefact.

Based on the above distinction, knowledge topics can be characterised in terms of their focus on, for example, functionality or quality in isolation, or their focus on bridging the gaps between purpose, functionality, and design in terms of design rationale.

- **Historical perspective**: Given an artefact with a design, one may consider different versions of this artefact's design over time.

  In the case of a system, one may consider the current version, the version that will be in existence after the development project has finished, and the (sketchy) version of the 'future' system that serves as a navigational beacon in a sea of possibilities to guide future development. In the case of a development process, one may consider the execution plan/strategy that is being used at the moment, or the plan/strategy that was used before.
- **Abstraction level**: When considering a domain, one may do so at several levels of abstraction. Various forms of abstraction can be distinguished: for example, type-instance, generalisation/is-a, encapsulation, and hiding of implementation details.

As mentioned before, depending on their concerns, stakeholders may be interested in different knowledge topics. For example, a financial controller will be interested in an investment perspective of the overall scope of a future system, a designer will be interested in all aspects of the design chain from different perspectives, etc.

4.2.3 Explicitness of Knowledge

The actors in a system development community have a need to communicate system development knowledge among each other. In the field of knowledge management, a key distinction is made between *explicit* and *tacit* knowledge (Nonaka and Takeuchi 1991). *Explicit knowledge* refers to knowledge that can be externalised in terms of some representation. With representation of knowledge, we refer to the process of encoding knowledge in terms of some language on some medium, e.g., creating an architecture model.

However, not all forms of knowledge lend themselves well to explicit representation. For example, the ability to maintain one's balance on a bicycle is learned by (painful) trial and error rather than reading instructions. This knowledge is actively and personally passed on from generation to generation: parents assist their children in this process by encouraging

them and by protecting them from serious injury during the trial-and-error process. In Nonaka and Takeuchi (1991), this is referred to as *socialising* as a means to transfer knowledge that cannot be made explicit. The type of knowledge concerned, which cannot easily be represented on a medium, is referred to as *tacit knowledge*.

Our focus is on the communication of system development knowledge by way of explicit representations, in other words *explicit knowledge*. In the context of this book, these representations mainly take the form of architecture descriptions. As discussed in Sect. 4.1, our initial theoretical considerations cover development of systems in general. In accordance with this generalisation we will, for now, use the terms *systems description* and *system description language* rather than the terms *architecture description* and *architecture description language*.

System descriptions are essentially forms of explicit knowledge pertaining to an existing/future system: its design, the development process by which it was/is to be created, the underlying considerations, etc. Given this focus, we can make a more precise classification with regards to what we mean by 'explicitness'. Based on Franckson and Verhoef (1999) and Proper (2001), we identify the following dimensions of explicitness for representations of system development knowledge:

- **Formality**: The degree of formality indicates the type of language used to represent the knowledge. Such a language could be formal, in other words a language with an underlying well-defined semantics in some mathematical domain, or it could be informal – not mathematically underpinned, typically natural language, graphical illustrations, animations, etc.
- **Quantifiability**: Different aspects of the designed artefact, be it (part of) the target or the project domain, may be quantified. Quantification may be expressed in terms of volume, capacity, workload, effort, resource, usage, time, duration, frequency, etc.
- **Executability**: The represented knowledge may, where it concerns artefacts with operational behaviour, be explicit enough so as to allow for execution. This execution may take the form of a simulation, a prototype, generated animations, or even fully operational behaviour based on executable specifications.
- **Comprehensibility**: The knowledge representation may not be comprehensible to the indented audience. Tuning the required level of comprehensibility of the representation, in particular the representation language used, is crucial for effective communication. The representation language may offer special constructs to increase comprehension, such

as stepwise refinements, grouping/clustering of topically related items/statements, etc.
- **Completeness**: The knowledge representation may be complete, incomplete, or overcomplete with regards to the knowledge topic (see previous subsection) it intends to cover.

4.2.4 Transformations of Knowledge

During the development of a system, the knowledge about the system and its development will evolve. New insights emerge, designs are created, views are shared, opinions are formed, design decisions made, etc. These all lead to transformations of the 'knowledge state' of the development community as a whole. The transformations of this 'knowledge state' are brought about by conversations. This immediately raises the question: what are these 'knowledge states'?

The discussion above already provides us with some insight into the answer to this question. The representations and the actors in a development community can both be seen as *harbouring* certain *knowledge topics*. As such, both representations and actors are (potential) *knowledge carriers*. Knowledge topics refer to some sub-domain of the system being developed and/or its development process. The knowledge topics can therefore be classified further in terms of their focus, scope, etc., as discussed in Sect. 4.2.2.

The actual knowledge that is harboured by a knowledge carrier is not explicitly taken into account. The knowledge that is available from/on/in a knowledge carrier is a subjective notion. An aspect of this knowledge that we can reason about more objectively is its level of explicitness, as we have seen in Sect. 4.2.3.

The knowledge as it is present in a development community can be seen to evolve through a number of states. Knowledge first needs to be introduced into the community, either by creating the knowledge internally or importing it from outside the community. Once the knowledge has been introduced into a community, it can be shared among members of that community. Sharing knowledge between different actors may progress through a number of stages. We distinguish three major stages:

- **Aware**: An actor may become aware of (possible) knowledge by way of the sharing by another actor.
- **Agreed**: Once knowledge is shared, an actor can make up his or her own mind about it, and decide whether or not to *agree* to the knowledge shared.

– **Committed**: Actors who agree to a specific knowledge topic may decide actually to commit to this knowledge. In other words, they may decide to adapt their future behaviour in accordance with this knowledge.

There is no way to determine objectively and absolutely the level of awareness, agreement, or commitment of a given set of actors. It is in the eyes of the beholder. Since these 'beholders' are actors in the system development community, we can safely assume that some of them will be able to (and have a reason to) judge the level of sharing of knowledge between sets of actors, and communicate about this.

4.3 Conversation Strategies

The knowledge transformations as discussed in the previous section are brought about by conversations. These conversations may range from 'atomic' actions involving a small number of actors, via discussions and workgroups, to the development process as a whole. This has been illustrated informally in Fig. 4.1.

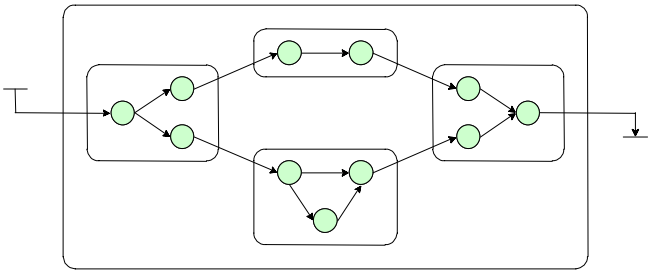

Fig. 4.1. Example sequence of conversations.

Each conversation is presumed to have some *knowledge goal*: a knowledge state which the conversation aims to achieve (or to maintain). In achieving this goal, a conversation will follow a *conversation strategy*. Such a strategy is needed to achieve the goal of the conversation, starting out from the current state.

Conversations take place in some situation that may limit the execution of the conversation. We may characterise such a situation further in terms of situational factors:

– **Availability of resources**: Refers to the availability of resources that can be used in a conversation. The availability of resources can be re-

fined to more specific factors such as time for execution, actors, intellectual capacities needed from the actors, or financial means.
- **Complexity**: The resources needed for the conversation, the knowledge being conversed about, etc., will exhibit a certain level of complexity. This complexity also influences the conversation strategy to be followed. Examples of such complexity factors, inspired by Franckson and Verhoef (1999), are the heterogeneity of the actors involved, the quantity of actors, complexity of the technology used, the complexity of the knowledge being conversed about, and the size of the gap between the initial knowledge state and the desired knowledge state.
- **Uncertainty**: If you want to determine a conversation strategy fit for a given situation, you have to make assumptions about the knowledge goal, the initial state, the availability of resources, as well as the complexities of these factors. During the execution of a conversation, some of these assumptions may prove to be wrong. For example, the commitment of certain actors involved may be lower than anticipated (initial state); materials needed for a workshop may not be available on time (resources); during a requirements elicitation session it may come to the fore that the actors involved do not (yet) have enough knowledge about the future system and its impact to formulate/reflect on the requirements of the future system (initial state).

  Note that it may actually be part of a conversation strategy to first initiate conversations that aim to reduce these uncertainties, in order to reduce potential adverse consequences.

If you formulate a conversation strategy, you should take all of the above-mentioned factors into account. A conversation strategy should typically cover at least the following elements:

- **Execution plan**: As we said before, a conversation can be composed of sub-conversations. Each of these sub-conversations focuses on a sub-goal, but they all contribute towards the goal of the conversation as a whole. The execution plan of a (composed!) conversation consists of a set of sub-conversations, together with a planned execution order.
- **Description languages**: The description languages to be used in the conversation(s).
- **Media**: The kind of media to be used during the conversation(s).
- **Cognitive mode**: The cognitive mode refers to the way in which knowledge is gathered or processed by the actors involved in a conversation. We distinguish two options:
  - *Analytical approach*: When information is processed analytically, the available information is simplified through abstraction in order to

reach a deeper and more invariant understanding. An analytical approach is typically used to handle complexity.

- *Experimental approach*: When using an experimental approach the project actors learn from doing experiments. The purpose is to reduce uncertainties by generating more information. Experiments can, for example, be based on prototypes, mock-ups, benchmark tests of migrated components, or other kinds of techniques which make the results of migration scenarios visible.

You may need to combine the two cognitive modes in specific situations, in particular in the case of conversations that are composed of several smaller sub-conversations.

- **Social mode**: The social mode is the way in which the actors executing the system development process collaborate with the actors from the business domain. We distinguish two options:
  - *Expert-driven*: In an expert-driven approach, project actors (the experts) will produce descriptions on the basis of their own expertise, and interviews and observations of business actors. The descriptions can then be delivered to the business actors for remarks or approval.
  - *Participatory*: In a participatory approach, the project actors produce the descriptions in close cooperation with some or all the business actors, e.g., in workshops with presentations, discussions and design decisions. A participatory approach may allow the acquisition of knowledge, the refinement of requirements and the facilitation of organisational change.
- **Communication mode**: We can distinguish a small number of basic patterns of communication here, as covered by combinations of the following five factors:
  - *Speaker–hearer ratio*: Most typically many to one, one to many, one to one, many to many.
  - *Response*: Simply whether or not an answer is expected from the hearer; if a response is indeed expected, one response may lead to a further response, leading to dialogue and turn taking.
  - *Time lag*: Whether or not communication takes some time between 'speaking' and 'hearing'. Consider the difference between a telephone call and an e-mail message.
  - *Locality*: Whether or not there is a perceived distance between participants. Note that this is a relative notion; two people communicating via videophone between Tokyo and Amsterdam may feel 'close', while two people from different departments housed in the same building may feel 'distant'. Distance can be not only physical, but also cultural.

- *Persistency*: Whether or not a message can be kept after communication, i.e., can be 'reread'. This is of course closely linked to the medium used, but it may also be related to the status of a document: persistency of a 'temporary document' or intermediary version may actually be counterproductive.

We can use combinations of these factors to typify many different modes of communication, which can have a major impact on the resources required for communication and the likelihood that a knowledge goal is reached. For example, one-to-many communication is relatively efficient and effective, assuming that no immediate (*n*:1) response is given; however, if a time lag is added, *n*:1 responses become possible but the one participant will have to invest much time to digest all these responses. Also, if *n*:1 responses are given rapidly, but the communication is persistent (e.g., people respond through altered copies of a file), then these responses are no problem except for the load on the recipient. And if many relatively distant people participate, in-depth and context-dependent communication will be difficult.

In a modelling context, not all combinations (communication modes) will be relevant, but it is still vital to consider things like 'Do I expect anyone to respond to this model?'; 'How many people will have to respond?'; 'How distant are they?'; 'How quick will the response (have to) be?'; 'How long will it take me to process responses?', etc.

A summary of this discussion is provided by Fig. 4.2. Given a *knowledge goal*, an *initial state*, and *conversation situation*, a *conversation strategy* can be determined, which should lead us from the *initial state* to the *knowledge state* as desired by the *knowledge goal*, taking into account the *conversation situation* at hand.

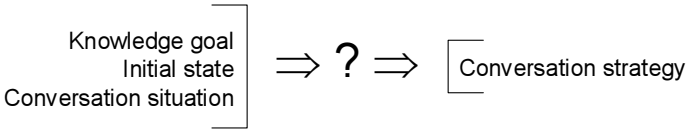

Fig. 4.2. From knowledge goal to conversation strategy.

4.4 Architectural Conversations

After the theoretical discussions of the previous sections, we now return to the practice of communicating enterprise architectures. The situation as depicted in Sect. 4.2 may indeed portray the underlying mechanics in the-

ory, but it still leaves practitioners with the question of how actually to produce such a conversation strategy. In all fairness, current research into these matters is still in its initial stages. The theoretical model as discussed above will have to be scientifically validated and refined. In addition, practical heuristics should be formulated, matching elements from conversation strategies to conversation situations and thus addressing the gap between the knowledge goal and the initial state.

Even so, we can already provide practitioners with some guidance in selecting conversation strategies to communicate about enterprise architectures, by reducing the discussion of selecting a conversation strategy to the selection of a class of architectural conversation in conjunction with an appropriate architectural viewpoint. To direct this selection, we will define a number of classes of architectural knowledge goals. The selected viewpoints identify *what* shall be conversed about, and what *language* (and language conventions) shall be used to do so, while the selected *conversation technique* identifies the style of conversation that is to be used.

So this section provides a discussion of the classes of architectural knowledge goals and conversation techniques that we distinguish within the context of enterprise architecture, as well as their relationship. In Chap. 7, the notion of viewpoint will be discussed in more detail, and additional heuristics on the selection of viewpoints and conversation types will be given.

4.4.1 Knowledge Goals

In Sect. 4.2.4, we identified three major stages in communicating knowledge: awareness, agreement, and commitment. Based on these and on the levels of sharing of knowledge and explicitness of knowledge as identified in Sect. 4.2.3, we can identify the following classes of knowledge goals that you may want to achieve in a conversation:

- **Introduction of knowledge**: This refers to situations where there is a need to introduce into or create new knowledge in a (part of a) development community. These kinds of knowledge goals typically lead to training or awareness sessions.
- **Agreement to knowledge**: With this class of knowledge goals, we refer to situations in which the mutual agreement of different stakeholders (with their own specific stakes and concerns!) needs to be improved or validated.
- **Commitment to knowledge**: In these cases, the knowledge goal goes beyond that of achieving agreement. Stakeholders should be willing to act upon the knowledge they agree to.

Note that the introduction of knowledge, as described above, may pertain to a subset of the development community. At the start of a system development project, the development team may not (yet!) have knowledge pertaining to the specific application domain. Domain experts and other informants, by nature of their roles, do have this knowledge. The development community as a whole comprises at least both the development team and the domain experts. A domain analysis session involving, for example, a business analyst and a domain expert *introduces* (part of) the domain knowledge of the domain expert into the development team.

4.4.2 Conversation Techniques

In architecture development, we find a number of common conversation techniques where it concerns the communication of architectural models:

- **Brown-paper session**: Structured brainstorm-like group session (up to about 15 people) in which items (keywords or short phrases) are elicited from the individuals in the group in answer to a question such as: 'What are the key functionality issues in our current IT architecture?' Typically, every individual item is written on a small adhesive note ('Post-It'). The items are then collected on a sheet of paper (traditionally of the cheap *brown* kind) and, by means of an open and creative group process, structured and categorised. This may involve adding, deleting, merging, or changing items. Usually, a mediator or facilitator is involved.
- **Elicitation interview**: An interview where an analyst puts informative questions to the informants. The aim is to gather knowledge from the informants. Interviews can be more or less 'open': they can be strictly focused or guided, but the conversation can also be left open to go where the interest of the interviewer or informants leads it.
- **Workshop**: Involves one to, say, fifteen people, working on a model or view interactively, mediated by an architect or analyst. This class also encompasses so-called joint modelling sessions. A popular, effective, and realistic technique is to project a view or model and have a facilitator adapt it in full view of the participants, thus generating immediate feedback. With a few participants, a workshop can of course simply take place behind a screen and keyboard.
- **Validation interview:** An interview where an analyst will aim to find out if the view or model matches the views and expectations of an informant. This could be a view or model that has been communicated to the informants beforehand, or during the interview. A validation interview will typically be much more 'closed' than an elicitation interview:

there will have to be some systematic approach by which validity of the view or model is checked.

- **Committing review**: A group of stakeholders are presented with a number of alternative models or views and their impact. They are requested to select one alternative and commit to this alternative based on their insights into the potential impact. This typically involves a formal decision-making processes (Franckson and Verhoef 1999).
- **Presentation**: Involves one to three people presenting a model or view to a group of, say, up to a hundred people. One may decide to elicit feedback, but this is usually gathered afterwards, in a more personal way, or at least 1:1 (e.g., through a feedback round).
- **Mailing**: A form of 'mass' communication, where a model or view is presented or handed over to a large number of people. Feedback may or may not be encouraged (feedback round).

Even though we have not yet discussed viewpoints, we can already relate the identified knowledge goals to the conversation techniques. This is shown in Table 4.1, which is based on interviews and discussions with many architects from industry.

Table 4.1. Knowledge goals and conversation techniques.

	Knowledge Goal		
Conversation Technique	Introduce	Agree	Commit
Brown-paper session	++	+	-
Elicitation interview	++	+	-
Workshop	+	++	+
Validation interview	-	++	+
Committing review	-	-	++
Presentation	++	-	-
Mailing	+	-	-

A + indicates that a certain conversation class is well suited for the selected technique of knowledge goals, while ++ indicates that it is particularly well suited. On the other hand, a - indicates that a certain conversation technique is ill-suited for the selected class of knowledge goals, while -- indicates that it is particularly ill-suited.

This table can fruitfully be used in practice to choose the conversation technique for the task and knowledge goal at hand. In Chap. 7, we will

have a more in-depth look at the use of viewpoints to assist communication between the different stakeholders.

4.5 Summary

In the previous sections, we have presented both a theoretical and a practical perspective of the issues involved in the communication of enterprise architectures. The theoretical perspective described the communication during system development in general. Based on the one hand on this theoretical view and on the other hand on the experiences of architects, the practical perspective presented a number of practical guidelines and conversation techniques that should aid architects in the selection and definition of architecture description approaches that are fit for a specific communication situation.