Information and Software Technology 56 (2014) 1233–1252

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Knowledge transfer, translation and transformation in the work of information technology architects

Mayara Costa Figueiredo [a], Cleidson R.B. de Souza [a,b,*], Marcelo Zílio Pereira [c], Rafael Prikladnicki [c], Jorge Luis Nicolas Audy [c]

[a] Universidade Federal do Pará, Belém, Brazil
[b] Instituto Tecnológico Vale, Belém, Brazil
[c] Pontifícia Universidade Católica do Rio Grande do SUL (PUCRS), Porto Alegre, Brazil

ARTICLE INFO

Article history:
Received 6 December 2012
Received in revised form 28 March 2014
Accepted 1 April 2014
Available online 12 April 2014

Keywords:
IT architecture
Software architecture
Architect roles

ABSTRACT

Context: Information Technology (IT) architects are the professionals responsible for designing the information systems for an organization. In order to do that, they take into account many aspects and stakeholders, including customers, software developers, the organization's business, and its current IT infrastructure. Therefore, different aspects influence their work.
Objective: This paper presents results of research into how IT architects perform their work in practice and how different aspects are taken into account when an information system is developed. An understanding of IT architects' activities allows us to better support their work. This paper extends our own previous work (Figueiredo et al., 2012) [30] by discussing aspects of knowledge management and tool support.
Method: A qualitative study was conducted using semi-structured interviews for data collection and grounded theory methods (Strauss and Corbin, 1998) [5] for data analysis. Twenty-seven interviews were conducted with twenty-two interviewees from nine different companies through four cycles of data collection and analysis.
Results: Companies divide IT architecture activities among different roles. Although these roles receive different names in different organizations, all organizations follow a similar pattern based on 3 roles: enterprise, solutions and software architects. These architects perform both the technical activities related to the IT architecture and the social activities regarding the communication and coordination with other stakeholders and among themselves. Furthermore, current tools used by IT architects lack adequate support for all these aspects.
Conclusion: The activities of the different IT architects are highly interconnected and have a huge influence in the way the requirements are handled in every phase of the development of an information system. The activities of IT architects are also important for knowledge transfer, translation and transformation, since they receive from and spread information to different groups of stakeholders. We also conclude that they lack appropriate tool support, especially regarding support for their collaborative work.

1. Introduction

The main goal of any information system is to attend to client and user requirements. However, clients and users are not always sure or do not even know what they want [27]. That is why requirement analysts are essential to the development of an information system. Nevertheless, there is still a considerable gap to be covered between requirements and a working system. In other words, during the construction of any information system, requirements need to evolve from client and user desires expressed in natural language into technical aspects and constraints developed by developers. In this process, aspects such as current and potential customers, organization's business, existing IT infrastructure, and others also influence the information system and need to be reflected in the working system. In fact, in large-scale companies software is seldom built from nothing; instead, it is embedded in

* Corresponding author at: Instituto Tecnológico Vale, Belém, Brazil. Tel.: +55 91 3213 5571; fax: +55 91 3213 5400.
E-mail addresses: mcfigueiredo@ufpa.br (M.C. Figueiredo), cleidson.desouza@acm.org (C.R.B. de Souza), marcelo.zilio@acad.pucrs.br (M.Z. Pereira), rafael.prikladnicki@pucrs.br (R. Prikladnicki), audy@pucrs.br (J.L.N. Audy).

http://dx.doi.org/10.1016/j.infsof.2014.04.001
0950-5849/

a large ecosystem that includes other information systems, organizational patterns and standards to be followed, costs and businesses goals it needs to achieve, among other factors [28]. While requirement analysts are responsible for understanding and eliciting client and user desires and software developers are responsible for the development of a particular software, *information technology (IT) architects* are the professionals in between these two groups. They take all the aforementioned aspects, and also the work of analysts and developers into account when designing an information system: they define the components that make up the information system of the entire organization instead of the components of a single system, and establish how products acquired and already developed systems are going to be integrated to compose the overall information system of the organization [7].

Given the importance of IT architects' work some companies and organizations have looked into ways to support it. For instance, customized modeling languages have been proposed [14]. There are also associations that seek to study and improve the career of IT architects (e.g., IASA [1] and The Open Group [2]). These associations created certifications trying to better regulate and create a common knowledge base to this career. According to them, IT architects occupy a broader role, while enterprise architects, software architects and other roles involved with architecture work are specializations of the first one. For this reason, we adopt this nomenclature (IT architects), given that it encompasses the other architecture-related roles.

Despite their importance, to the best of our knowledge, there are no studies focusing on the work performed by IT architects. Therefore, this paper goes a step beyond previous research by describing a qualitative study conducted with IT architects. As is usual in qualitative studies, our research question is very broad: *How do IT architects perform their activities in industry?* In order to answer this question we conducted 27 semi-structured interviews [24] with 22 different architects from 9 different system-developing companies. We used grounded theory methods [5] for data analysis. Consistent with results from the literature, our results suggest that there are 3 different types of IT architects who all perform similar roles in the organizational ecosystem at different "points" in the organizational hierarchy. More importantly, our results extend the literature by suggesting that those architects work as "bridges" between the customers (and analysts) and the developers, transforming [29] the business requirements and the organizational constraints into technical aspects that can ultimately be implemented by software developers. To be more specific, each architect is a "bridge" between different set of actors in such a way that the activities of the different IT architects are highly interconnected. Based on data from the interviews and an analysis of tools, frameworks, and modeling languages suited for IT architects, we also suggest that IT architects lack tool support, especially regarding collaborative aspects that can help them better coordinate their work given their interdependencies [27]. The work reported in this paper is an extension of our earlier work described in [30].

The remainder of the paper is organized as follows. Section 2 presents background information about IT architecture and the different types of IT architects. Sections 3 and 4 describe the setting of the study and the research methods used respectively. Section 5 presents the findings followed by their implications in Section 6, while Section 7 discusses tool support for IT architects. Finally, Section 8 presents our conclusions and suggestions for future work.

2. IT architecture

2.1. Architect roles

IT architecture is defined by IASA as "the art or science of designing and delivering valuable technology strategies" [6]. This definition emphasizes that IT architecture is not focused on delivering solutions or projects in a particular timeframe. Instead, it focuses on delivering technology *strategies*, of which solutions and projects are only a part. More specifically, IT architecture defines the components that make up the overall information system of an organization. It generates a plan that defines how acquired products and developed systems will be integrated to compose the overall information system strategy of the organization. It also enables the management of IT investment in a way that more clearly meets business needs [7].

The professional responsible for developing the IT architecture is the *IT architect*, who defines strategies for solving customer business problems or needs through the use of information technology. Those strategies include systems, applications, processes, hardware and software components, and the integration of many kinds of products, technologies, and services [8].

Based on the aspects discussed above, it is no surprise to find out that there are many different aspects that an IT architect needs to be able to handle. In fact, some authors [9–11] argue that this role is considered "ambiguous and murky" because it interacts with many different types of stakeholders and each of them expects the architect to work in a different way. According to these authors, an IT architect needs to be, at the same time, (i) a generalist, when she/he interacts with managers for example; and (ii) a specialist, when she/he interacts with software developers or other technical personnel. In order to reduce this ambiguity many companies establish different types of architects, also called IT architect roles [11]. However, these different roles are not standardized, i.e., each company establishes the roles it considers most adequate.

In a first step towards standardization, IASA and The Open Group discuss disciplines or specializations of the IT architect role [6,8]. The idea is very simple: every IT architect should have the same basic knowledge and then acquire a higher level of proficiency in one IT architecture discipline or specialization. Examples of disciplines include: software architecture, information architecture, business architecture, technological infrastructure architecture, network architecture, etc.

In this paper, we will use the definition presented by the IASA, which suggests four different architect roles [9] (described below). It is important to highlight that this definition was used to describe our results, but did *not* guide our research as a preconceived theory or view. This will be described in detail in Section 4.

- *Enterprise architect*: this architect works to support the organization's business strategy with IT solutions and information. He/she is responsible for the overall IT strategy and ensures that the IT architecture is cost effective, i.e., that the IT investments are aligned to the organization's business strategies. He/she is responsible for strategies at many different levels, such as global standards and strategies related to security and overall infrastructure. This requires a deep knowledge of business, IT, enterprise architecture, business modeling, governance, project management and economy experience; in addition to leadership and negotiation skills. Akenine [9] argues that this role is similar to a city planner who, using strategy, planning, and regulations, is responsible for different functions in a city that must work together effectively.
- *Business architect*: this role also focuses on IT solutions for the entire organization. The business architect focuses on the organizational business needs and understands in details how the organization works. He/she is usually involved in business related areas and also in process improvement efforts. This architect suggests improvements in the organization IT department together with enterprise architects, once they understand how information systems support the organization's business.

Business architects help in process modeling and support solution architects with analysis and requirements of new or existing solutions. They have a deep knowledge of the business, process modeling and requirements analysis. Thus, although they have an organizational focus, they are also active in ongoing projects working to ensure that these projects deliver benefits to the organization's business [9].

- *Solution architect*: this architect focuses on the ongoing projects and works in designing IT solutions based on requirements from the organization business. He/she aligns new solutions with architectural principles, considering standards, architectural principles and integration within the organization, and taking into account the reuse of existing organizational capabilities. Solution architects are responsible for balancing functional and non-functional requirements, defining priorities and trade-offs. Their goal is the success of the current project. They have broad and general technical knowledge, and competencies in infrastructure, data models, service orientation, and enterprise architecture [9]; and
- *Software architect*: this role focuses on the ongoing project similarly to solution architects. While solution architects have a wider focus on policies, regulations, and reuse of existing assets, software architects have a deeper knowledge in technology (opposed to business). They work in designing and structuring software systems, dealing with both functional requirements and quality attributes, such as reusability and performance. Some quality attributes are obviously shared with solution architects. Software architects must have deep knowledge on programming languages, frameworks, standards and technical modeling [9].

It is possible to note that as the activities of an architect get closer to the definition of low level components and to technical aspects (UML, programming languages, structuring classes, etc.), they also get closer to what software engineering researchers and practitioners usually calls software architecture. The work of software architects has not been sufficiently explored in empirical studies with the exception of the work described by Smolander [3,33] and Grinter [4]. However, based on what has been presented above, the activities of software architects comprise only one part of the entire IT architecture process, i.e., a software architect is only a particular type of IT architect. To the best of our knowledge, this work is the first one to address the work of IT architects.

3. Settings

This paper describes a study that aims to understand how IT architects perform their activities in industry. To do that, we conducted research in system development organizations, focusing on the roles that perform architectural activities and on their relationships. Although we could identify a set of possible activities performed by IT architects based on a literature review, this approach would be too limited because it would constrain our data collection to aspects already reported in previous studies. Since we wanted to understand how IT architecture is developed in practice in a real organization, we adopted a qualitative approach, given the exploratory nature of the study. Thus, we visited 9 different IT organizations where we conducted interviews with professional architects and other related roles. The companies studied and reported on this paper are described below and summarized in Table 1. Information in these descriptions and table was collected from the companies' websites and through a small questionnaire sent to the interviewees. This questionnaire had questions related to each column in Table 1, but not all interviewees responded to the questionnaire and, among those who responded, not all were aware of the requested information (e.g., size of projects). We also tried to collect information in the companies' websites, but they also did not have all the necessary information for filling the entire table. We decided to provide all the information we had in order to characterize the companies the best we could, although this information does not provide a direct comparison. The description of each company is thus not standardized and there is some missing information in Table 1. Company names have been anonymized due to confidentiality issues.

Company A – This is a multinational company headquartered in the United States with many branches around the world, working with distributed software development. It is an IT company that has a broad portfolio of products, from personal customers to large companies, offering complete solutions: services, hardware and software. It has millions of customers in the world and 170,000 employees.

We conducted our research in the Brazilian branch located in the city of Porto Alegre. The division where we collected data has a SW-CMM level 2 certification and, according to our informants, its processes are CMMi level 3, although they have not yet paid for the evaluation. The company develops hundreds of projects per year and its projects usually last from 6 to 12 months. This company is generally involved with large projects: for instance, one of our informants said that he is currently working on a project with around one million lines of code.

Company B – This is a multinational IT company headquartered in the United States. It develops, sells and supports computers and related products and services. Among these products are personal computers, servers, data storage devices, network switches, software, and computer peripherals. Company B has many branches around the world and employs more than 96,000 people worldwide. We started to research (in the first iteration of the study) in a Brazilian branch located in Porto Alegre, but we were not able to return due to commitments of our interviewees, so we could not get as much information as we would like to in our research.

Table 1
Companies summary.

Company	Clients number	Employees number	DSD	Projects per year	Projects size	Employees per project	Projects duration	Maturity Model
Company A	Millions	170.000	Y	Hundreds	1 Million loc	70	6–12 Months	CMMi 2
Company B	Millions	96.000	Y	–	–	–	–	–
Company C	10	150–200	I	50	100 Use cases	5–15	1000–2000 h	CMMi 3
Company D	32	140	I	130	–	5–15	3–4 Months	CMMi 2
Company E	Millions	–	Y	–	–	–	–	N
Company F	1	500*	N	1**	–	–	5 Years**	N
Company G	1	230	N	1**	17,000 Classes	230	5 Years**	N
Company H	200	1600	Y	40	–	20	4 Months	N
Company I	15	–	I	–	–	–	–	CMMi 2

* Company total employees number. We researched only one division.
** Company has only one project that is either not finished (Company G) or is an inside project (Company F).

Company C – This is a Brazilian software development company, specialized in e-commerce for the retail industry. Its headquarters are located on a Technology University Campus, in Porto Alegre, Brazil, and it also has offices in Portugal. This company provides IT services of different areas of expertise to medium and large organizations. Currently, according to our interviewees, the company has from 150 to 200 employees, 10 customers (half in Brazil and half in Portugal), and usually develops around 50 projects per year, each, depending on the project type, having on average 100 use cases and between 1000 and 2000 h. Depending on project size, it usually has between 5 and 15 employees working on each project. It was evaluated at CMMi level 3.

Company D – Company D is a Brazilian IT consulting and systems development company headquartered in Porto Alegre, Brazil, with another office in São Paulo, Brazil. It has existed since 2002 and provides services in software development and infrastructure through consulting, project factory, outsourcing and offshoring. Company D has operations in Brazil and overseas assignments, including customers in segments of energy, agribusiness, information technology, transportation, logistics, petrochemicals and retail. According to our interviewee, this company develops around 130 projects per year and has 32 customers; 30 in Brazil and 2 abroad. It has CMMi level 2 evaluation.

Company E – Company E is a multinational Internet company with headquarters in Spain and operates as a web portal and/or an Internet access provider in the United States, Spain, and many Latin American countries. We did research at one of the Brazilian affiliates, which is the head office of Latin America.

Company F – This company develops corporate solutions for credit cards, with products for HR and fleet management (supply and maintenance). It works especially using the Internet as a tool for self-management, enabling its customers to control limits, request and cancel cards over the web, etc. The company has around 630 employees in 9 different states in Brazil. This company has its own IT department and we interviewed 2 architects from this area. This area is a software factory that provides services to the company. The IT area thus has only one customer, the company itself.

Company G – This Company is a Brazilian state-run provider of IT solutions focused on the banking sector. It is headquartered in Rio de Janeiro, Brazil, and is controlled by a Brazilian state bank. Products and services include integrated software solutions and outsourcing, network management and technical support. Company G has many affiliates in Brazil; we did research at the affiliate located in Belém.

Company H – This company is a global IT consultancy headquartered in the United States. It delivers custom application and provides consulting. It also has many affiliates around the world. We did research at a Brazilian affiliate, located in Porto Alegre. This affiliate has only been open for 11 months and we were not able to get as much information as we would have wanted for our research.

Company I – This is a Brazilian Company headquartered in Porto Alegre, Brazil. It is focused on database and web software technologies, development and implementation of corporate and critical-mission applications. It has development in Brazil and also in the United States and Portugal through partnerships with other companies. Company I has CMMi level 2 evaluation.

Table 1 summarizes the companies studied. As previously explained, there is some missing information because not all interviewees responded (or knew all the answers) to every question. Column DSD indicates if the company works with distributed software development: "Y" is for yes, "N" is for no, and "I" is for initial, which means that the company has already worked with DSD but it was either a pilot project (Company C) or it is not usual for the company (Companies D and I).

4. Research methods

4.1. Qualitative research

Our goal in this research was to understand how IT architects perform their activities in industry. Considering the characteristics of this goal, our research was conducted as a qualitative study. Qualitative research tries to understand the experience of a determined group of people. It is research that produces results that cannot be achieved through statistical procedures or similar methods [5]. The main benefit of this type of research lies in the fact that qualitative methods compel the researcher to delve deeper into the complexity of the problem instead of abstracting it. Therefore, the results are richer and more informative, helping to answer questions involving variables that are difficult to quantify, such as human characteristics like motivation, perception and experience [31].

4.2. Data collection and analysis

As for data collection methods, we used semi-structured interviews [24]. Semi structured interviews have an intermediate control level, and are composed by a mix of closed and open questions, making it possible to collect the kind of information the researcher is seeking and also the unpredicted kinds. [31]. The interview presents a conversation flow and new questions can be developed in real time, as the researcher learns new information [31].

The data analysis method used was Grounded Theory (GT) [5]. We chose GT because we wanted to understand architectural activities as they are really performed, instead of descriptions from books or scientific papers. In order to do that, it is important to obtain the information from people who actually perform the activities, in our case using interviews, and to systematically analyze it. GT allows the creation of a theory derived from the data, systematically collected and analyzed by a research process, so GT is useful and appropriate to achieve our goal.

In addition, as in other qualitative studies, in GT there is no *a priori* theory; the researcher starts with a broad research question that can be refined during the study. Our research question fits this characteristic (How do IT architects perform their activities in industry?) and we decided to explore architect activities without any preconceived theory, addressing many different aspects of architecture and software development, along with broad questions that aimed to instigate the interviewees to freely speak about their work. As our work progressed, we focused on additional aspects including the different types of IT architects, their interactions among themselves and with other stakeholders, and finally, the extent to which their current tools supported their work.

GT also strongly recommends that data collection and analysis iterations be intertwined. The analysis starts as soon as one has a significant amount of collected data and this analysis will direct the next data collection [5]. Furthermore, the next data collection is based on the concept of theoretical sampling: it consists of choosing the points of the next data collection to expand, complement or deny the theory that is being constructed. It is done to explore the different conditions in which the concepts vary [5].

GT is based on codifications (open, axial and selective codifications [5]), which consist of assigning a category or a concept to a piece of data. Therefore, a theory is a set of categories well developed and systematically related among each other in order to form a theoretical framework that explains the phenomenon studied [5].

Considering these aspects, our initial research question was very broad, and focused on understanding how IT architects perform their activities in industry, in real organizations. To try

Table 2
Data collections summary.

Company	Interviewees and roles	Data collection
A	Interviewee 1: Technical architect	1, 2 and 4
	Interviewee 14: Technical architect	4
	Interviewee 15: Architect	4
B	Interviewee 2: Technical leader	1
	Interviewee 3: Developer	1
C	Interviewee 4: Architect	1, 2 and 4
	Interviewee 5: Architect	1, 2 and 4
	Interviewee 16: Architect	4
D	Interviewee 6: Distributed projects manager	1
	Interviewee 17: Architect	4
E	Interviewee 7: Technology coordinator/ architect	1 and 4
	Interviewee 8: Technology manager	1 and 4
	Interviewee 18: Architect	4
F	Interviewee 9: Architect	2 and 4
	Interviewee 10: Architect	2
G	Interviewee 11: Architect	3
	Interviewee 12: Architect	3
	Interviewee 13: Architect	3
H	Interviewee 19 and 20: Systems consultants	4
I	Interviewee 21: Architect	4

to answer this question we interviewed employees who worked on IT architecture activities. Our focus was on architects, but we also interviewed other roles in order to acquire a broader knowledge of subject (the interviewees' roles are listed in Table 2). This also allowed us to gather information about architectural activities that were performed by roles not directly labeled as architect. This will be discussed later in the paper.

Overall, we conducted 4 iterations of data collection, which were always intertwined with data analysis. In total, we conducted 27 interviews with 22 different informants. Interviews were recorded and lasted from 23 min to 1 h and 27 min. Overall, we have 19 h of interviews that consist of more than 295 pages of transcribed data. Table 2 summarizes the companies and interviewees from each data collection. Relevant quotes are presented in the results and were freely translated into English by the authors.

4.3. The process of analysis

The study started from "scratch," without any theoretical model guiding data collection and analysis. However, it integrated existing research results in the last two phases of data analysis, as described below. In addition, in the first cycle of the study we looked for companies having distributed software projects because this kind of development accentuates the difficulties of software development [32]. However, as we conducted the interviews and analyzed the data we realized that aspects of our results were not exclusive to distributed software development,[1] so we decided not to limit the company sample in the following cycles.

4.3.1. The first cycle

In the first data collection we conducted 8 interviews in 5 different organizations (Companies A to E) located in Porto Alegre, Brazil. The companies (in this and in all iterations) were contacted through convenient sampling, given that one of the authors has contacts in the Technology Park of his University where several software companies are located.

The author who had access to the companies made the first contact asking for employees who worked with software architecture and could help in the research. We next contacted these employees and scheduled the interviews. The interviews were conducted in a meeting room and we asked to record them, to which all interviewees agreed. We also provided a confidentiality agreement in which we assured we would not disclose confidential information or names to anybody but the researchers. During the interviews, we took notes of any interesting aspect that was mentioned.

The interviews of this iteration focused on the different aspects that influence software architecture development, emphasizing the importance of architects' work especially in the context of distributed software development. In order to conduct these interviews we developed an interview guide[2] that was divided into the following sections:

(a) general questions (i.e. name, years in the company, role);
(b) questions about distributed software development;
(c) questions about the daily work of the interviewee, asking them to use a current project as example whenever possible;
(d) questions about the software architecture; and
(e) questions about the interaction among the different teams involved in the construction of the information system.

The questions were created by the authors based on the research question. In addition, during the interviews, we were free to explore aspects unanticipated by the guide as the informant mentioned something interesting to be discussed.

The interviews were transcribed and analyzed using the MaxQDA tool [19]. We analyzed the data line by line to identify units of meaning and coded them into categories and subcategories using the open coding technique [5]. MaxQDA allows the linking of an amount of text into a category and it also has means to easily report and return every quote linked to each category of all documents used in the research. The category names were based on the terminology used by the informants and on the terms contributed by the researchers. Examples of categories are: architecture activities (which has "technical" and "non-technical" as subcategories) and task division. After the open coding, we started the axial coding to understand the relationships in our data.

During analysis of this first iteration we learned about the different architect types through our interviews. As mentioned before, our interview guide had questions about the interaction among teams and roles and, we also always encouraged the interviewees to explain how interaction happened and which were the roles they interacted the most with. While answering this question, one interviewee directly mentioned that her/his company had three different types of architect called enterprise, solution and technical architect. Thus, during the open coding a category (named "Architect Types") was created to represent this information. This interviewee described the activities each type of architect performed in his(her) company. During the axial coding, we compared this information with that grouped in the "architecture activities" and "task division" categories and realized that there seemed to be a pattern: every company seemed to have a division of tasks related to architecture.

The constant comparison method was employed with the data and categories and relationships identified among the concepts. This process was applied cyclically in order to help us proceed to the next iteration more focused and therefore able to ask additional questions (e.g., about the different architect types), listen and observe in more sensitive ways. Thus, based on the analysis of the first iteration, we planned the next data collection cycle.

[1] We argue that our results concern distributed and collocated software development.

[2] The interview guides are presented at Appendix A.

4.3.2. The second cycle

From the results of the first analysis we elected some aspects to be refined in the next iteration of data collection and prepared the second collection accordingly, i.e., we performed theoretical sampling (see Section 4.2). The main aspect we decided to refine was the division of architecture related tasks. We conducted interviews with new interviewees and follow-up interviews with interviewees from the first iteration. More specifically, we conducted 5 interviews in 3 companies (follow-up in Companies A and C, and two new interviews in Company F). Data from those interviews were integrated to the same categories of the first set and analyzed together. We performed open and axial coding again (also cyclically performed) with the data from this iteration.

At this point, we had around 30 categories encompassing many different aspects, such as division of labor, architect's activities, collaboration, DSD difficulties, and awareness. Another example is the category named "Architecture", which had 12 sub-categories (Who, When, How, Description, Distribution, Modules Interaction, Boundary Objects, Difficulties, Different Visions, Technology, Importance, and Liaisons). This category aimed to gather as much information as possible about software architecture. Other category names were: "Division of Labor" (sub-categories "Model", "Locations", "Distribution"), "Architect Activities" (sub-categories "Technical" and "Non-Technical"), "Architect Types", and "Interaction among teams". We first categorized all interviews according to these categories, and then we analyzed all the data. During this second data analysis, we noticed one recurrent aspect: the presence of different roles participating in the architecture activities and the interaction among them. We also observed the relationship of these architectural roles with aspects of information diffusion, communication, and coordination during the development process of information systems. We thus again refined the focus of our research based on our data, as suggested by grounded theory [5]. We noticed that companies not only have different architecture roles, but these roles are organized in such a way that the development of the information systems these companies create is facilitated. We categorized these roles based on the activities they perform and on their interactions with other roles. Considering this, we identified three main roles, and provisionally used the names Company A gave to their roles, since all of the other roles seemed to fit into them. These architecture roles play an important function by transforming [29] knowledge about customers and organizational requirements into aspects that can be implemented by software developers.

At this point we discovered the IASA study describing the different architect types. We also conducted a literature review of this aspect to understand what was known about it. We found that there were only a few studies about this, and they were mainly performed by organizations such as IASA and The Open Group. We decided to adopt the nomenclature suggested by IASA, because it was based on a research with many other industries. We then elected this aspect, and all the information and knowledge flow related to it as the leading aspect of our research. We should emphasize that, to the best of our knowledge, this information flow is *not* covered by studies from IASA or any other author.

Based on the data from these two initial data collections, we started the process of selective coding. This is done through the selection of a core category and the integration and refinement of the other categories [5]. In our case, we created a new category called "Transformation",[3] since we observed that as the software system is developed, information is transformed from customer and organizational requirements into technical aspects. We also reorganized our categories to integrate and refine them considering the new main category and, consequently, we revisited our interviews and previous codes. For example, the prior category called "Architect Types" was transformed into a broader category called "Architect" with many sub-categories. One of them was named "Roles" and divided into three sub-categories, one for each architect role we identified. All the categories were reorganized in order to better describe this transformation process, which is performed by the different IT architects. This process is explained in more details in Section 5.

4.3.3. The third cycle

In the following data collection, we conducted 3 interviews in a *new* organization located in Belém, Brazil, again using theoretical sampling, i.e., choosing data collection points to expand and/or test the theory under development [5]. We also added more questions about the interdependencies among the IT architects as well as their interactions. Our goal was to continue the process of selective coding, i.e., the refinement of our theory.

Once again the interviews were transcribed and coded and the data from this iteration was integrated into our previous datasets. We again conducted a literature review regarding IT architecture and architects. The analysis from our datasets and the results from our literature review guided our next data collection. After doing this, our theory was better defined, and we thus planned the next data collection focusing on theory validation.

4.3.4. The final cycle

Another aspect of selective coding is validation of the theory [5, pg. 159]. According to Strauss and Corbin [5], one way to validate the theory is telling it to the interviewees and asking them how the theory "fits" their particular cases. So, in the fourth data collection iteration we tried to interview all previous interviewees to validate our results. This is also known as *member checking* [36]. In order to keep our data from being restricted to a single's person point of view, i.e., not being representative of the entire organization, we contacted at least one *new* interviewee from every organization in this fourth iteration as another form of validation. We also conducted interviews in new organizations to verify if our theory would apply to them. Overall, we conducted 11 interviews with 5 old interviewees (Companies A, C and E) and 9 new ones (Companies A, C, D, E, F, H and I). On some occasions, we had more than one informant per interview. As in the previous iterations, these interviews were transcribed, analyzed and integrated into our previous datasets, contributing to the creation of an audit trail that explains our analysis [36]. The following section presents our results of this entire process of data collection and analysis.

5. IT architecture in practice

The main goal of any information system is to attend to client and user requirements. However, clients and users are not always sure or do not even know what they want [27]. That is why requirement analysts are essential to the development of an information system. Furthermore, a large information system is seldom isolated; often, it needs to be integrated into a set of other information systems from the organizations, and must adhere to its standards, protocols and constraints. In other words, the set of client and user needs and/or desires needs to be "transformed" into a (formal or informal) specification that can be implemented by software engineers. The professionals who work in this transformation process are the IT architects. Furthermore, these IT architects can be classified into 3 different types according to their "location" in the organizational hierarchy. To be more specific, each type of architect is a

[3] In our original analysis, the name of this category was Translation. Later on, we identified the work of Carlile [29] in transferring, translating and transforming knowledge and we then decided to use the term transformation which is more adequate for what we are describing in this paper.

"bridge" between different sets of actors in such a way that the activities of the different IT architects result in the specification to be implemented. In addition, IT architects also need to interact with one another passing along information relevant for them. In other words, their work is highly interconnected.

Fig. 1 is a representation of this transformation process, in which each architect mainly interacts with a set of stakeholders in addition to grouping and bridging their information (already grouped and transformed by him/her) to the next architect, who does the same thing with another set of stakeholders, until the information arrives at the software developers at the abstraction level they can interpret and work on. It is through this flow that the information system is developed.

In the following sections we describe each one of these aspects of the transformation work of IT architects. First, we present the different types of IT architects we identified. Next, we illustrate how each type of architect interacts with a very specific set of actors. After that, we discuss the interdependencies among the different architects. We conclude this section discussing how current IT architects' tools provide limited support for their transformation and collaborative work.

5.1. The architecture roles in practice

In the nine companies studied, we identified that the division in roles can be represented in the structure presented in Fig. 1. This result was based on all the information linked to our categories. One of them was called "Architects" and was detailed in many sub-categories during open codification. Among its sub-categories were: "Roles" (also expanded into in "Enterprise", "Solution", and "Software"), "Interaction among architects", "Interaction with other stakeholders", and Collaboration. During axial coding, excerpts from the interviews contained in these sub-categories were linked, together with other excerpts from other categories (for example, the category "Good Practices"). Then, after the selective coding, we created Fig. 1, which summarizes the transformation process. This figure illustrates how each architect role receives information from different groups of stakeholders, performs his/her own work using this and other information and then passes a different type of information along to the next architect. As discussed before, previous work [8,9,11] suggests a division of the IT architecture activities in different architecture roles. Our results are thus aligned with previous research, but go further by analyzing the interactions and information flow among these architect roles.

We related the companies' architect roles with the ones described in Fig. 1 based on the activities each role performs. In the first analysis we identified the activities the architects performed, and separated them among technical and non-technical (category "Architect activities" and its sub-categories "Technical" and "Non-technical"). We detailed this analysis according to the activities performed by enterprise, solution and software architects. During this analysis, because we noticed the pattern of activities associated with each architect role, we associated information from other categories with them, especially the ones related to collaboration, interaction, and information diffusion (we had sub-categories named "Collaboration", "Interaction among architects", "Interaction with other stakeholders", and "Information diffusion"). Fig. 1 was developed based on this analysis.

It is important to emphasize that some of the organizations observed have the architect roles described in Fig. 1 formally defined and institutionalized, while in others these roles are informal. In some organizations these roles are *not* called architects, despite performing architecture-related activities. In addition, our analysis did not identify the IASA role of the business architect in any of the studied organizations. What we observed was that when there is an enterprise architect, this role encompasses business architect activities. When the company does not have an enterprise architect, the solutions architect is the person responsible for performing the activities of the business architect, in addition to his/her own activities.

Fig. 1 represents the generic structure that we identified. For instance, we noticed a common aspect with all companies: there are always actors that perform activities from the solutions and software architect roles. However, the enterprise architect might or might not exist. More interestingly, each organization "adapts" this generic model according to its own interests. Despite the adaptation, we observed that the way companies organize the roles can be classified into 3 different groups according to definition of roles, namely: "defined roles", "partially defined roles", and "non-defined roles". Each one of these groups is summarized in Table 3 and described in details below, i.e., this table presents how each company maps these three groups of IT architects into its own roles. Column 1, named "Group" presents the names of the three groups of companies we identified according to how they define their roles. Column 2, named "Company" presents company aliases. Column 3, "Role Name", presents the roles identified in each company, while column 4, named "IASA Role", maps the roles in Column 3 with the ones suggested by IASA. For example, the first item in column one is the group called "Defined Roles". In this group we classified two companies, A and E (as identified in column 2), which means that these two companies have well-defined roles, namely enterprise, solution and technical architect (column 3). Column 4 indicates how these roles in company A are mapped into IASA roles: Enterprise Architect and Solution Architect are called the same, but Technical Architect in Company A is equivalent with what IASA calls a Software Architect.

In the "Defined roles" group there are well-defined roles for the enterprise, solution and software architects. They receive different names, but we identified that they perform activities that are

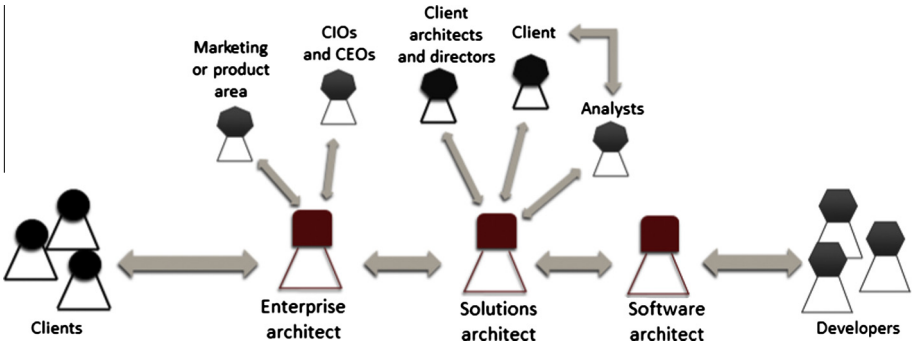

Fig. 1. Architect roles and the generic structure of their main interactions.

Table 3
Architect roles mapping.

Group	Company	Role name	IASA role
Defined roles	A	Enterprise Architect	Enterprise Architect
		Solution Architect	Solution Architect
		Technical Architect	Software Architect
	E	Functional Responsible	Enterprise Architect
		Technician Responsible	Solution Architect
		Team or Technical Leader	Software Architect
Partially defined roles	C	–	Enterprise Architect
		Software Architect	Solution Architect
		Designer	Software Architect
	D	–	Enterprise Architect
		Software Architect	Solution Architect
		Senior Developer	Software Architect
	I	–	Enterprise Architect
		Software Architect	Solution Architect
		Technical Leader	Software Architect
Non-defined roles	F	–	Enterprise Architect
		Software Architect	Solution Architect
		Software Architect	Software Architect
	G	–	Enterprise Architect
		Software Architect	Solution Architect
		Software Architect	Software Architect

similar to these three IASA architect roles (based on the "enterprise architect" category). Moreover, it is also possible to identify some typical activities of an IASA business architect among the activities that the enterprise architect performs. The two companies researched in this group are multinational organizations that also work with distributed development.

In the second group ("Partially defined roles"), the role of solution architect is defined by the organization, but not necessarily with this title: this role receives different names in different organizations but they have the same activities, especially those related to the definition of the overall solution architecture. In this group it is also possible to identify some activities performed by what IASA calls an enterprise architect, but with no role dedicated to it: the actor playing the role of solution architect is the same actor who performs the enterprise architecture activities in addition to the activities of a Solution Architect. The quotes related to these actors' activities were related to "Enterprise Architect" and "Solution Architect" categories at the same time. Because it was the same actor who performed all of them, we could identify that this person performed activities of both roles, but solution architect activities prevailed. Companies C, D and I are in this group. All have more than 100 employees and develop more than 50 projects per year. Some of these projects involve distributed customers and/or teams, but there are few such projects, or they are simply pilot projects or experiments; i.e., distributed development is not consolidated in these companies.

Finally, the last group ("Non-defined roles") comprises Companies G and F. In this group we were unable to identify organizational architecture activities clearly differentiated among the different architect roles (there was no interview excerpt assigned to the "Enterprise architect" category for these companies). On the other hand, solutions and software architects activities are well identified, but these activities are interchangeable among the architects (they usually do not have different roles' names; all are called "architects"). In other words, they are not performed by a defined role in every project and not even the person who performs the role is previously assigned. The person (or sometimes more than one) in the architecture team who identifies him/herself with the project solution architect activities assumes this job. The others automatically perform Software Architect activities. Another characteristic of this group is that the architects (both solution and software) have little contact with customers. Instead, the so-called analysts are responsible for this activity. The architects thus handle the requirements and if they need to discuss something about the requirements they have to talk to the analysts.

Note that Companies F and G have some similarities between them. They both call themselves software factories and have only one customer apiece: Company F is the IT division of a credit card solutions organization, while Company G is a regional branch of a state-run organization created to support a very large local customer. Both of them usually work in one project at a time. Their architecture team is composed of four architects who are simply called software architects. In each project there is therefore one architect who works and creates the overall architectural solution, working as a solution architect while the other architects assume typical IASA software architects activities, working closer with developers and developing critical components. According to our informants, this arrangement emerged during their daily activities.

5.2. The interdependencies between architect roles and other stakeholders

In this section, we describe how the division of labor according to architecture roles serves two different purposes. First, it is a way to organize the architects' work regarding their IT architecture activities. And second, it is at the same time a way to structure IT architects *communication activities*.

The first purpose, to organize the IT architects' work, is based on a better definition of the attributions of each role. For example, in the following quote, Interviewee 8 describes when and how the role similar to the enterprise architect in Company E works. Interviewee 8 works in a department where this role equivalent is called systems architect (unlike other departments where it is called "Technician Responsible", as defined in Table 3). During the coding activities, we assigned this quote to the "Enterprise architect" category, because it presents aspects that are part of the IASA enterprise architect activities. In the end, given that the quotes about this system architect role were assigned to this category, we concluded it was equivalent to the IASA enterprise architect. The same happened to the role called "Technician Responsible" at the same company. According to Interviewee 8, the systems architect does not effectively participate in the whole project, being more involved in the beginning of it. He has already held this role and defines it as follows:

> "He has more time to study new technologies; he has more time to get to know Company E's structure, to know other products *(...)* and he talks much more about making proposals, performing analysis *(...)*. He/she is not there to define the interaction with external products from the company, he is there to thoroughly detail what needs to be done in the project: 'we are going to use this component', 'there is already a free software product that makes this feature, we just need to adapt or extend it', etc." – Interviewee 8, Company E.

Consequently, since the enterprise architects (and similar roles) work focuses on organizational view, their activities are more focused on the beginning of the projects, in order to outline how this project will fit into the organization IT architecture. Considering this, the quote above also illustrates how this "context" in which

the new products are being built influences the work of the architects: in this case, it means that some software components might be used or adapted in this project.

This observation is also true about a different company; C. Company C does not have a specific role responsible for performing enterprise architect activities. However, we identified that the role equivalent to the solution architect also performs enterprise architect activities, along with his/her own activities (quotes related to this role were initially assigned to both enterprise and solution architect categories, although in later analysis we concluded that this quote was a better fit to the solution architect category). Despite the fact that this role mainly performs solution architecture activities, it is called "software architect" in Company C (as described in Table 3). In the following quote, Interviewee 4 describes this role. It is possible to identify the influence of the "context" described before, which is stronger in enterprise architect roles, when Interviewee 4 mentions investment and commercial relationships, information this role receives at the beginning of the project, when the company closes the deal.

> "The *[solution]* architect has a more separated function from all that: he gets the overall project idea, what the project is going to be. He defines that he understands which level of investment is going to be made in the project, and it does not depend on the analyst, it depends on the commercial relation. We receive some information at the time the contract is made: this customer is adopting Microsoft as a platform, and they are already buying this and that, and they are already setting up an environment... Meaning that we already know some information and the analyst has not started working yet, but we already know. So it is more with that information that we work. So, this person designs, based on all these ideas, how the system concept is going to be and plans what I said: a concept view, then a logical view and so on. He does not hold himself in development. He defines how the pieces of the puzzles are going to be connected, which is the most suitable product for each situation. And he still can review it later." – Interviewee 4, Company C.

This same quote also illustrates the organization of the other IT architect roles. The role described in it is equivalent to IASA solution architect and aims to propose the overall IT solution to be developed, but there is no need to worry about the specifications or diagrams. This is the responsibility of the technical leader in Company E and designer in Company C, both similar to the IASA software architect. This technical leader or designer is the actor who works closely to developers and focuses on more low-level aspects of the architecture. This information, assigned to the subcategory "Interaction among architects", illustrates the information used to develop the interconnections among architects in Fig. 1.

Thereby the division of labor organizes the IT architecture activities among the different roles: each role has its responsibilities better defined (even when it accumulates some of them) and works with a reduced amount of information. This information is provided both by the other architect roles and also by other stakeholder groups. This information flow is the other aspect benefited by the division of labor. It also serves to structure communication activities (as described in Fig. 1): each architect interacts mainly with a specific group of stakeholders. For instance, a solution architect needs to take into account the work products of the enterprise architect and also receives information from stakeholders such as directors and customer architects. On the other hand, software architects receive the work product of solution architects and interact mainly with developers. Interviewee 1 from Company A summarizes this communication aspect in the following quote (This quote is also assigned to the "interaction among architects" category):

> "What changes is the stakeholders' 'level' with whom each [*IT architect*] interacts. The enterprise architect talks more with CIOs and CEOs. The solution architect will talk with an architect from the other side *[company]* or with directors from the other side. And the technical architect interacts more with *[software]* developers. *(...)* And they communicate among each other". – Interviewee 1, Company A.

Examples of the interactions of different architect roles and other stakeholders were assigned to a different category, called "Interaction with other roles", in order to better understand all the interactions involved in the architects' activities. These interactions are illustrated in the following quotes. The first one describes the interaction between the marketing area and the technical architect (similar to a software architect), while the second quote involves the solution architect, stakeholders of the company service delivery area, and the development team, that in this case includes the software architect.

> "The marketing area brings its requirements. It is an entity in itself and has a list of requirements that it wants to meet, because the person from marketing *[who interacts with the software architect]* is attentive to what is happening on the market, especially what is happening in the other companies, so he can point out and say 'I am not managing to sell this product because the competitor has this functionality and he nullifies me with that, so I need something to end up with this advantage over him'. So that is what the marketing bring to us *[software architects]*". – Interviewee 1, Company A.
>
> "We have these three roles – service delivery, development team [including the technical leader, equivalent to the software architect] and the *[solution]* architect – to get the requirements, put them on a board and start to draw little boxes, how it works, how to solve it; I *[as a solution architect]* identify which components are going to be reused, which are going to be developed and among these ones we detail inside *[the boxes]* how they are going to work. *(...)* When there is a list of requirements, the *[solution]* architect is a piece of the puzzle, as well as the service delivery and the development team *[including the technical leader, equivalent to the software architect]*. He/she is not above or between them, but together with them. They are complementary roles. I defend this idea of complementary *[tasks]* instead of serialization *[of tasks]*". – Interviewee 7, Company E.

It is important to emphasize that the communication among IT architects and the different stakeholders is not always straightforward. There are communication problems related to the domain knowledge, and, IT architects devise strategies to handle these problems and get their work done. In the quote below, we describe a simple strategy based on a category called "Good Practices", and related to the interaction categories: a meeting held specifically to make sure people involved understand and use the same terminology:

> "A common language among the project participants is necessary. ... I mean: it is not the developer talking some technical language, it is not the requirement analyst who only speaks the business language and then the architect between then. It is necessary to have a specific project phase where common terms are defined (...) So, meetings for term alignment are needed" – Informant 16, Company C.

In addition to communication, architects also need to be able to negotiate with the different groups involved. If this negotiation does not happen, it can lead to problems in the project. These kind

of problems were identified during the analysis based on the relations among the categories "Interactions among architects", "Interactions with other stakeholders", "Problems and difficulties", "Good Practices", and "Bad practices". The following quote illustrates an example of this, where Interviewee 2, from Company B, describes a project, where she/he worked:

> There was a project that I worked in which I joined after the definition of the architecture, I 'tuned in' the architecture, and there was this distributed ... development team. They didn't like the architecture (...). It was really conceptual; it was not based on frameworks used in the market (...). It was pure architectural concepts applied, and it was good. I don't know if the other team had this visibility, but I went there and they said to me: 'do you know the system [X]? We are thinking of changing it to use Spring, what do you think? We would have to completely abandon the other architecture and adopt dependency injection'. Then I asked: 'But why do you want that?', 'We want it in order to better maintain the code. We would like to better maintain the code and to write less infrastructure code'. Because the prior architecture had more code to maintain, but since it was done and was stable I just needed to insert the business rules, but I might eventually need to maintain the architecture core. On the other hand, the team's decision to adopt a consolidated framework was to avoid this kind of problem, to use something that is already approved by the market as something stable, scalable etc., to make coding easier. So, you see: there were two teams in different locations that had different visions. One team wanted to create its own market independent architecture, while the other wanted to use a framework from the market which was already proved as good and working. And this second team proceeded and recreated the whole code and a new architecture was adopted. Both [architectures] worked.

In this case, there was thus one architecture that had been defined by the previous architect and that was market independent. This architecture was supported by one team. However, there was another team who wanted to change this architecture because they wanted to use a framework from the market. In this case, the solution architect was *not* able to negotiate with the different teams, therefore the architecture was changed.

5.3. The interdependencies among different architect roles

The categories "Interaction among architects" and "Interactions with other stakeholders" were compared and related with the categories of each architect role and the "Collaboration" category during axial coding. During selective code, they were linked to the main category ("Transformation") as a way to describe the theory we were developing. Based on that, we observed that IT architects interact among themselves in addition to interacting with very specific sets of stakeholders. Their activities are interconnected; they depend on each other's work in different ways. In summary, they receive information from different groups of stakeholders, perform their own work using this, and additional information they seek, and then pass a different type of information along to the next architect. The following quote from Company A illustrates this aspect.

> "This person *[enterprise architect]* talks about requirements, but in an open way, nothing really defined. The requirements are like 'I think we need to increase product scalability' and she/he does not say what in scalability we are going to increase, or how we are going to do it *(...)*. But she/he elaborates *[the requirements]* together with the customers *(...)* and she/he says 'It is important to have a greater scalability' and then the people 'below' him/her *[the solution and software architects]* can understand what we are going to increase *[an activity from the solution architects]* and how we are going to do it, which is the technical architect *[i.e., the IASA software architect]* part, where I work. So, this is the difference: the enterprise architect works at the higher level and she/he looks basically to how the industry is, and with that she/he is going to propose and seek solutions, pointing directions... So she/he sees market tendencies and says 'everybody today is working with VMWare; virtualization is important, so we must increase virtualization support', then the person here *[the solution architect]* is going to say 'yes, this is true, there is a large number of clients who demand it' and we go in this direction. Then, the other person *[the software architect]* refines the focus. This is the division according to the scope: the enterprise architect is broader and less specific, while the technical architect *[the IASA software architect]* is the lower level and is focused on the specific product; she/he is going to see how this product attend the directions that the higher level person *[the enterprise architect]* points. *(...)* There must be an alignment between these two dimensions in order for the product to be really agreed upon and from this point on to create a product roadmap where the main enterprise architect is focused on guaranteeing that what she/he directed is correct, validating it with customers *(...)* while the others *[the solution and software architects]* are focused on really developing the product to successfully fulfill this agreed roadmap." – Interviewee 1, Company A.

Therefore, enterprise architects develop the architectural principles [9] that will guide all projects in the organization (based on their interaction with a very specific and high-level group of stakeholders, e.g., the CEO and CIO). These architectural principles are very important to the solution architects, since they need to follow these principles to produce the architecture of each specific project of the organization. Thereby, the solution architect designs the overall solution of one project and then, this solution is divided in order to be developed. Each "part" of the overall architecture is refined by one software architect that works closely aligned with software developers. To be more specific, the following quotes, assigned to "Interactions among architects", "Solutions architect", and "Software architect" categories, are examples of how the solution architect influences the work of the software architect:

> "He *[the solution architect]* is not going to model a system, he is a person who wants to know where it came from, where it goes, where it fits in the world, why my customer is or is not going to use it, which are the high level use cases. He/she is hanging out with the customers, saying where the possibilities are, and he just says to us 'this is what needs to be done'." – Interviewee 1, Company A.
>
> "He [the solution architect] is floating with the customers [i.e. directors or architects from the customer] saying where are the possibilities and he just shouts to us [the technical architects who are equivalent to IASA software architects]: 'this is what must be done'. Then, I am a technical architect, I do everything we learn in college, like use cases, diagramming, ...". – Interviewee 1, Company A.

Finally, software architects need the overall architecture definition to perform their own work, i.e., to refine the architecture and guide developers through its implementation. In fact, it is not unusual fort one project to have more than one software architect as discussed below:

> "When a contract involves many areas, i.e. it involves a server part, a storage part, a network part and so on, each of these parts has a solution architect. They work focused for example in the storage part that is huge by itself. Then, inside the storage part there are other technical architects *[IASA software*

architects], who work within the products that will form a niche represented by the storage area of the project". – Interviewee 1, Company A.

Considering these interdependencies, it is possible to notice that the different architect roles still need to collaborate even with the division of labor (which explains the multiple categories assigned to some quotes, "Collaboration" included). In other words, in addition to having to manage the dependencies between their artifacts, it is also important to highlight that the different architecture roles work closely together. This means that the description of the work performed by the different architect roles does not represent a waterfall model or a hierarchy; but instead, different architect roles collaborate with each other. This situation is clearly illustrated in the following quote where Interviewee 1 highlights the interdependencies that exist in the work of the different architect roles:

"In a solution developed for an *[external]* customer, the enterprise architect will be involved. As we *[technical architects]* are closer to the development team, we are responsible for developing products that have a lifeline. So, since this lifeline was created, there is a parallelism among the 3 architects *[enterprise, solution and technical]*, because there is someone looking for marketing tendencies, there is another one taking care of the portfolio of that specific situation, and a third one who is worried about developing everything agreed on with the "superior" architect *[the enterprise architect]*. So, there is a parallelism *[in the tasks]*, but the three *[architects]* are working together and have contact points" – Interviewee 1, Company A.

These "contact points" mentioned by Interviewee 1 are usually meetings where all architects, program managers, managers of the business units, and members of the marketing team participate as a core team to contribute to the decision-making (this information was thus assigned to the category called "Good Practices"). These meetings take place many times during the project, especially at the milestones. So, despite the fact that each architect role has his/her specific activities, they need to interact considerably.

5.4. Tool support for architects' work

One aspect that we observed with special attention was the usage of tools by different IT architects. Therefore, we first had one category called "Tools" to assign all related excerpts, but then we noticed the need to create a different category for assigning excerpts about documentation, which was named "Documentation". During axial coding it was possible to relate both categories, and then in selective coding we associated this relationship with our main category according to our theory. Broadly speaking, our results suggest two aspects. First, the observed organizations tend to use a mix of tools, usually very simple ones. And second, the documentation produced in these organizations tends not to be very formal. In the rest of this section, we will focus on aspects that illustrate the problems faced by IT architects because the tools they use do not support important aspects of their work.

In the first quote, Interviewee 1, a software architect, explains how she/he passes information to the requirements team. She/he speaks in the same quote both about tools and documentations, already indicating the relation among the categories. In this interviewee's opinion, it is more important to transfer information in a standardized way to the development team, than for this information to arrive standardized to him/her, especially concerning requirements. If the information transfer to the development team happens in a standardized way, this facilitates this team's understanding of the information and eliminates some of the "burden" of the architect, who has to explain in details what the requirements mean. To address this issue, his/her team created a template for the requirements and used a tool to manage them in his/her current project.

"In the project we are working on today it is a bit better *[compared with previous projects]* because we dedicated a certain time to standardizing the requirements and creating a template for requirements. And we started to use a tool for requirements management to improve this ... and the template to improve the understanding of the requirements. We have had some improvements, but we are still far from what we would like. Today the development team still has to suffer considerably to better understand the things they receive." – Interviewee 1, Company A.

In the literature there is some evidence regarding the use of tools to facilitate the spread of information within software development teams. As an example, Lopes et al. [37] suggested a strategy to document requirements using natural language based on English. The idea behind the proposal was that distributed teams could use an English-based template to document the requirements and use this globally using the same template.

In the context of Interviewee's 1 project, as mentioned, they use both the template and the tool for requirement management to help in the transformation process, making the information easily understood by the development team. However, as Interviewee 1 argued, this is not enough, which led us to relate these two categories to the "Problems/Difficulties" category.

Another issue raised in Company A comes from the fact that this is a multinational company with employees performing different architect roles spread around the globe. It is therefore really important for them to have mechanisms to support these architects' activities, especially their interdependencies, given the geographical distance among them. Interviewee 1 mentioned the tool used for requirements management:

"We are using *[Tool A]* here *[at Company A]*. This tool imposes a work flow chart that I will use. There is already a process for controlling requirements and project changes; this tool helps us with that. It helps to create a requirement inside it and makes it easier for remote teams to review these requirements. It is a big issue for the *[Tool A]* us *[the remote teams for doing online reviews] (...)*. The ideal is to have more offline work." – Interviewee 1, Company A.

"For me this is a key point to this tool *[Tool A]* achieve success: to be able to adapt to the project reality. And if the project is distributed the tool needs to support it" – Interviewee 1, Company A.

According to the interviewee, this tool has the potential to facilitate distributed reviews producing more effective meetings: people (not only architects) would go to meetings knowing the issues to be discussed. This would provide a better use of participants' meeting time, given that they are distributed and it is not easy to bring them together into a meeting (especially the architects). If people go to meetings unprepared, important aspects might not be properly discussed and could re-appear later in the project causing rework and delays. It also shows the relation of "Tools" and "Documentation" categories with the "Collaboration" category. Furthermore, in the literature one can find several studies regarding the use of tools to support requirements review sections. For example, in the work of Calefato et al. [38], the authors developed a tool called eConference and have conducted several experiments in order to evaluate the feasibility of the tool for distributed teams.

However, Interviewee 1 also points out that people lack the motivation to use this tool (again showing the relation among "Tools" and "Documentation" categories and "Problems/Difficulties"

category). So, although this tool has the potential to help in several aspects, it is not always effective. On the other hand, Company C has adopted a tool that stores the documentation and provides traceability among the different types of documentation, as illustrated in the following quote:

> "The idea is that all the project documentation should be stored within Tool C, divided by sections. And as the project evolves, the types of diagrams, the models evolve too, always keeping the connections *[i.e., the traceability]* in order for the developer to be able to go from his class to, for example, a non-functional requirement that is implemented by that class" – Interviewee 4, Company C.

Despite the usage of Tool C, all interviewees consulted in this company confirmed they still have problems with requirements; mostly problems concerning changes that cause rework (which strengthens the relation among "Tools" and "Documentation" categories and "Problems/Difficulties"). As happens in Company A, revisions and a better use of meeting time would help to avoid this problem, information that was recovered from the "Good practices" category.

Another interesting point that we observed is that most of the organizations we studied tend to use common documents and wikis to create and maintain the different "architectures" (especially the software architecture) and the associated documentation. This is illustrated in the following quotes (assigned both to "Tools" and "Documentation", but also to "Good Practices").

> "Basically, we use a flowchart where we have the system behaviors. It is composed of macro boxes that are opened later in another similar diagram, detailing the logic of the program operation *(...)*. If something is not clear in the diagram, we put it in a document. And we use a wiki *(...)*. To me it is documentation that is easy to do and to understand and is really useful. *(...)* I do not like to be stuck with a tool: we use wiki, we use power point and we use *[Tool E]*. We use a white board with pictures too, which is really good for documenting many things". – Interviewee 7, solution architect, Company E.

> "We have a tool for project management called *[Tool D]*. *[Tool D]* works integrated with risk management, a wiki and other tools related to project management. We have customized what we call trackers, which are similar to bug tracking tools. We have trackers for architectural aspects, for architectural requirements. We get this documentation and store it in the wiki. Our artifact is the wiki". – Interviewee 18, solution architect, Company D.

As mentioned before, Company D is in the "Partially defined roles" group, which means that Interviewee 18 is a solution architect who also performs enterprise architecture activities, while a senior developer performs the activities of a software architect (see Section 5.1). The trackers mentioned by this interviewee are used by both the solution and the software architects (showing the relation between "Tools" and "Interaction among architects" categories). For instance, if a new architectural requirement arises, the solution architect performs his/her activities (related to an overall definition of how this new requirement affects the overall architecture) and uses the tracker to delegate the task (related to the requirement refinement) to a software architect, who is going to perform his/her tasks and them pass it along to the development team.

The use of wikis and common documents is well documented in the literature. For example Prikladnicki and Carmel [39] found that companies usually use several tools for collaboration. This includes screen sharing, project management and configuration management tools, global repositories, continuous integration and bug reporting tools, and wikis.

Some companies use more specific tools, but usually either this tool does not attend to all their needs or the tools need to be customized by the organizations and be used alongside additional tools. For example, company D uses Tool D that is a project management application which, among other features, presents a flexible tracking system. This company has adapted this tracking system to support architecture activities (among others), as described in the quote:

> "We have customized what we call trackers, which are similar to bug tracking tools. We have trackers for architectural aspects, for architectural requirements". – Interviewee 18, solution architect, Company D.

Meanwhile, Company E uses Tool E, which is a diagramming tool, *not* a tool specific for IT or software architecture. Company C uses Tool C, which is a tool for architecture. The specification of this tool claims it has features related to requirements management, business process modeling and enterprise architecture frameworks (among others). However, as mentioned before the interviewees of this company also complained about problems with requirements, changes and rework.

5.5. Summary of key findings

This section briefly summarizes our key findings, introducing aspects that will be addressed on the next section. Our first result is the fact that all studied organizations adopt a well-defined division of labor, i.e., there are different types of IT architects. These organizations formalize these types of architects in different ways: in some organizations the roles are well-defined, while in others they are non-defined, informal and *ad-hoc*. Second, each IT architect performs both technical and non-technical (mostly communication and coordination) activities. Third, each type of IT architect interacts with a particular set of stakeholders and as a consequence deals with different types of knowledge. Fourth, our results also indicate that IT architects collaborate among themselves because their work is highly interdependent, i.e., the work of each IT architect influences and is influenced by their colleagues. This means that IT architects spread information they acquire from a specific set of stakeholders to other IT architects. In other words, the different IT architects are responsible for the communication flow and information diffusion within the organization. In so doing, IT architects help the organization to achieve a common understanding about the project. This understanding is not trivial since the information that flows in the organization might come from or be directed at different stakeholders, therefore being misinterpreted. IT architects avoid misunderstandings by "processing" the information they receive from another IT architect or stakeholder in such a way that this information becomes comprehensible by other architects and/or stakeholders. Finally, given the importance of a common understanding about the IT architecture, one would expect that the IT architects would have different tools to support their work. However, our last result suggests that current tools used to support the creation and management of IT architectures provide limited support for the way the IT architects work. All these aspects are discussed in the following sections.

6. Discussion

6.1. It architecture and knowledge management

As we illustrated in the previous section, the process of developing a large information system involves different stakeholders with different requirements, interests and priorities. In such a complex scenario the presence of many different organizational

boundaries that need to be navigated is not a surprising. In particular, we are interested in *knowledge boundaries*, as discussed by Carlile and Rebentisch [26]. Knowledge boundaries are boundaries created when there are different and specialized knowledge domains and the individuals of these domains need to interact and exchange information [26]. For instance, Carlile and Rebentisch illustrate this problem by discussing how engineers from different groups (styling, engine and power-train, climate control and safety) need to interact to design a new vehicle. Furthermore, there are different "types" of boundaries. In a *syntactic* boundary, the differences and dependencies among actors are known and the knowledge is not so specialized. In this scenario, a common syntax or knowledge is sufficient to specify the differences and dependencies, and then the information can simply be *transferred* across the boundary. A *semantic* boundary occurs when different knowledge domains generate differences in interpretation, thus creating discrepancies of meanings. In this case besides a common syntax, it is necessary to establish a common semantic in order to identify and *translate* these different interpretations and dependencies among the different knowledge domains. Finally, when the knowledge domains are so specialized that they generate different interests, building common knowledge to be shared among the groups becomes a political process. This is an example of a *pragmatic* boundary. In this case, both groups need to adjust and *transform* their current way of performing their activities and their knowledge in order to accommodate the knowledge from the other group and thus to collaborate in the boundary. In other words, to learn, one needs to transform his/her own knowledge [26].

Fig. 2 illustrates the idea of knowledge domains in the context of our dataset. More specifically, Fig. 2 (based on Fig. 1) portrays how each architect role receives information from different groups of stakeholders, performs his/her own work using this and other information and then passes a different type of information along to the next architect. Based on our analysis presented in the previous sections, Fig. 2 also identifies the boundaries between the different knowledge domains. These include syntactic, semantic, and pragmatic boundaries.

The first knowledge domain, the *organizational*, includes actors such as CIOs and CEOs and embraces knowledge about planning, contracts, business deals and partners, i.e., management. The marketing experts bring knowledge about the market, competitors, and market tendencies. These two aspects of management and marketing have the same focus: the organizational aspects that influence the creation of an IT solution. In the *functional* domain are included business specific knowledge and focus on the functionalities that the information system needs to implement. This information is acquired by analysts (e.g., requirement analysts) who need to deeply understand the clients' domain. In other words, both analysts and clients are in the *functional* knowledge domain. Finally, the knowledge domain in which the software developers are embedded embraces programming languages, tools, frameworks, development patterns, etc. In other words, this domain embraces *technical* knowledge.

These three knowledge domains represent the main specialized domains in the development of information systems, therefore they also indicate the knowledge boundaries that need to be crossed in IT projects, i.e., in order to fully develop a software system it is necessary to combine knowledge of these different domains. As we can observe in Fig. 2 each domain has a set of actors associated with it, *but also an associated IT architect*: the enterprise architect in the organizational domain, the solution architect in the functional one, and the software architect in the technical domain. These architects are the ones who combine the knowledge of these domains for the successful development of an information system. Using Carlile's [29] terminology, IT architects *transfer*, *translate* and *transform* knowledge across these boundaries. Furthermore, each architect is responsible for "managing" the boundaries of a knowledge domain according to the type of IT architect. This is done by sharing information and collaborating with the IT architect from the other knowledge domains in such a way that the important information from one domain (e.g., the marketing team) is *transferred*, *translated* or *transformed* into something meaningful for another domain. An example of *transferring* knowledge can be identified when Interviewee 1 from Company A mentions that they standardize the way their requirements are presented using a template. This was mentioned in Section 5.4, but we will repeat part of it here:

> "It is a bit better in the project we are working on *[compared with previous projects]* because we dedicated a certain time standardizing the requirements and creating a template for requirements. And we started to use a support tool for requirement management to improve this management and the template to improve the understanding of the requirements."

Another example of *transferring* was mentioned at the end of Section 5.2 when we discussed the meetings held specifically to make sure different stakeholders understand and use the same terminology. In both cases, the idea is to create a common syntax among the different stakeholders to specify the differences and dependencies among them, and by doing so, allot these

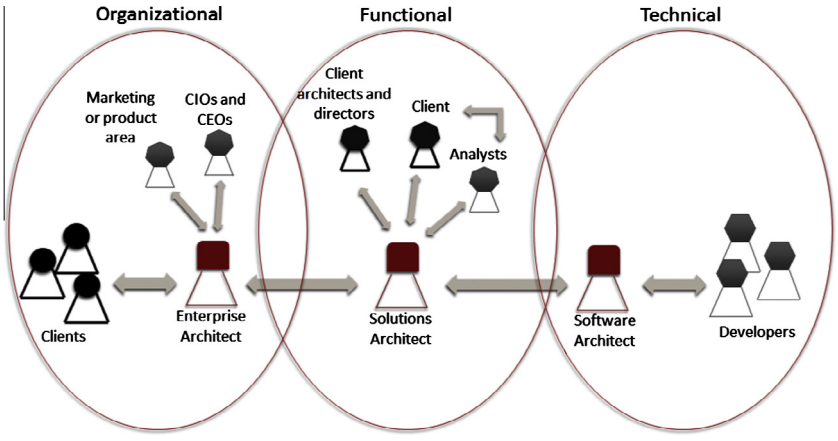

Fig. 2. Knowledge spheres and the architect roles.

stakeholders work together and knowledge can be *transferred* across the boundaries.

However, this is not enough: IT architects also need to *translate* knowledge across boundaries. The translation occurs, for example when, the enterprise architect aggregates information from customers, marketing experts and other stakeholders to make sure she/he has the "right" information that is then passed along to the solution architect. This is clearly represented by the quote of Interviewee 1 presented in Section 5.2, and repeated here:

> "What changes is the stakeholder 'level' with which each [*IT architect*] interacts. The enterprise architect talks more with CIOs and CEOs. The solution architect will talk with an architect from the other side *[company]* or with directors from the other side. And the technical architect interacts more with *[software]* developers. *(...)* And they communicate among each other". – Interviewee 1, Company A.

Each architect thus *translates* the knowledge of the different stakeholder groups inside each main knowledge domain represented in Fig. 2. However, it still is necessary to share knowledge through the three main domains, but they have become too specialized and a simple translation is no longer enough. It is also necessary to *transform* knowledge across the boundaries. IT architects perform this activity when they interact with each other, negotiating and passing along the knowledge of their domain. The quote from Interviewee 1 repeated above illustrates this when he says "and they communicate with each other". The negotiation aspect of the knowledge transformation is also illustrated in the following quote excerpt, repeated from Section 5.3:

> "(...) This is the division according to the scope: the enterprise architect is broader and less specific, while the technical architect is the lower level and is focused on the specific product; she/he is going to see how this product attend the directions that the higher level person *[the enterprise architect]* points. *(...)* There must be an alignment between these two dimensions in order for the product to be really agreed upon and from this point on to create a product roadmap where the main enterprise architect is focused on guaranteeing that what she/he suggested is correct, validating it with customers *(...)* while the others *[the solution and software architects]* are focused on really developing the product to successfully fulfill this agreed roadmap." – Interviewee 1, Company A.

With this process, a common knowledge can thus be created and shared through the whole project. If this transformation (and the negotiation within it) does not occur, it can generate some problems in the project as discussed at the end of Section 5.2. The quote, by Interviewee 2 from Company B, illustrates how knowledge from the functional and technical domains was so specialized that it needed to be negotiated, i.e., there was a pragmatic boundary. However, the teams in these domains were not able to negotiate a single software architecture and, in the end, had two different architectures to solve the same problem. We argue that if both teams had adjusted and transformed their knowledge this kind of problem would have been avoided.

6.2. It architecture and software architecture

At this point, it is important to discuss our results in the context of previous research into the work of IT and software architects. To begin with, we need to mention the very relevant work of Grinter [4]. She performed a qualitative study about software architects and concluded that these professionals act as "boundary spanners" helping to create and maintain the common knowledge every development project needs. Our research extends her work by considering the overall IT context. In other words, what Grinter observed for software architects is also true for other types of IT architects. In addition, we explain how IT architects transform knowledge from one domain to another so that it is possible to develop an information system in large organizations taking into account constraints, standards, patterns and other technical and organizational aspects.

Similarly, Unphon and Dittrich [23] describe a different qualitative study, in which they report that the (software) architecture usually exists implicitly in *discussions* during software development, with *no* huge effort in documentation. They report that the software architect acts as a "walking architecture": she/he communicates the architecture to developers and, in turn, communicates problems to the right stakeholders [22]. In short, they illustrate how software architecture in practice emphasizes communication rather than documentation.

We argue that this paper extends the results of these two studies. First of all, our data illustrates that every IT architect needs to work as a "boundary spanner," not only software architects. Each IT architects needs to handle communication and coordination within his/her knowledge domain: because architecture activities are divided, the associated interactions and communication are also divided so that each role interacts with different groups of stakeholders. Second, the different IT architect roles need to collaborate among themselves to get their work done. Third, in every company we studied we observed the importance of communication rather than documentation, not only for software architects, but for IT architects as well. This means that these companies had several "walking architectures", instead of a single one. Finally, the lack of documentation and appropriate tool support are mentioned by the informants as problematic.

Another related work was discussed by Smolander [3,33] who conducted interviews with what he regarded as the most central roles in software development: designers, architects, managers (project, department, technology, and process development managers), and customers. Smolander identified four metaphors for software architecture that are used by the different stakeholders: architecture as blueprints, as literature, as language and as decision. He also identified that the meanings related to architecture varied according to the position and background of the person interviewed, and so argued that different groups of stakeholders use different metaphors. For instance, designers often use the blueprint metaphor, but do not even recognize the language one. Smolander then concluded that these metaphors implicate in "many and even mutually conflicting perspectives of architecture to be described" and that these conflicts should be considered for any development process and method used in organizations [3]. He then argued that architecture should be approached as a "vehicle for achieving common understanding" [33].

Although Smolander's focus was on software architecture, his conclusions and our data allow his results to be extended to IT architecture as well. For instance, software architects are closer to software implementation, so the blueprint metaphor is more related to them; the architecture as a decision metaphor is more related to management roles, so it is more useful to enterprise architects, since they are closer to the high management of the company; and the architecture as literature metaphor focuses on the reuse of components and because of that can be related to both solution and/or enterprise architects. Finally, the architecture as language metaphor is important to all IT architect roles, because one of their main tasks is to guide the transformation of customer needs into technical aspects that developers can effectively use in their work.

All the previous works are related to studies of software architecture in practice. However, it is also important to relate our work

with that of requirement analysts, especially in how they manage non-functional requirements. For instance, Ameller et al. [34] studied software architects through interviews. Their results suggest that the role of software architect was *not* formalized in the 13 companies studied: the person who performed software architects' activities was chosen according to his/her technical ability and experience in the project domain. In addition, these "architects" could not identify their activities as software architecture activities because they overlapped with other activities they performed. In addition, non-functional requirements (NFRs) were mainly elicited by these "architects". In 10 out of the 13 projects discussed in the interviews, the software "architect" was the main source of the NFRs. In the three other projects the clients elicited NFRs. These projects were all outsourced projects, and even with clients working in the NFR elicitation the architects also played an active role in completing the definition. Ameller et al. [34] also identified communication problems concerning the meanings of words, especially with regard to the definition of types of NFRs. We believe this is a problem related to the syntactic boundary, when knowledge needs to be transferred across this boundary. Finally, they also found that NFRs were often not documented, and when they were, the documentation was not precise and usually became desynchronized. In other words, Ameller's and colleagues' results are similar to ours. They reported aspects of communication (instead of documentation), knowledge transfer, and the important work of software architects in identifying (non-functional) requirements. In addition to these results, we also report how the work of the software architect is related to the work of other IT architects and the *translation* and *transformation* work conducted by different IT architects.

To summarize and conclude this section, our results suggest that IT architects act as boundary spanners [4], they work along the boundaries of knowledge domains by facilitating knowledge transfer, translation and transformation among different group of stakeholders. This is achieved by a division of labor in which IT architects have different roles, and each role is located in a knowledge domain. By so doing, each architect role is a "walking architecture" [23] and can be associated with a particular software architecture metaphor [33].

7. Supporting it architects

As previously mentioned before (see Section 5.4), we were interested in understanding the tools used to support IT architects. In this section we discuss this aspect in more details.

Based on what our interviewees argued about tool support, we performed a literature review to identify some of the tools available for IT architects. We wanted to understand whether current tools would provide adequate support for the work of the IT architects. We identified several tools and strategies that aim to support their work. For instance, there are initiatives such as the Zachman Framework for Enterprise Architecture [13], which is a two-dimensional classification scheme that provides a structured way to represent an enterprise [16], the EPBE (Eriksson–Penker Business Extensions), a UML extension to enable the use of UML for modeling business [14], and the BPMN (business process management notation), which is a flowchart-based notation for specifying business processes [15].

The problem with the aforementioned approaches is that they focus on one (or only a few) type(s) of architect(s). When tools focus on more than one level, this is done without taking into account the interconnections among them [18]. There are, however, other approaches that aim to support the different types of architects in a more integrated fashion. The TOGAF framework [17] is an example of this case. It is an architecture framework that "provides methods and tools for assisting in the acceptance, production, use, and maintenance of an enterprise architecture". The phases of this framework seem to encompass activities of all main architect roles, because they support architecture activities at both the organizational and technical levels (with focus on ongoing projects).

A different approach presented by Schekkerman [18] is an architecture tool selection guide that compared many architecture tools verifying their compliance with aspects such as governance, risk, program management, business/IT strategy, among others. According to his results, tools such as Alfabet Planning IT and IBM-TeleLogic System Architect Family + Rhapsody seem to attend to most of the mentioned aspects, i.e., they provide some level of support for all four architecture roles. On the other hand, tools such as Sparx Systems Enterprise Architect and IBM Rational Software Architect seem to be more related to solution and software architecture activities [18].

Despite that the availability of different types of tools, in the field we observed a great usage of very simple tools such as wikis.[4] We noticed that wikis provide an easy mechanism to allow access to and changes in the architecture (including its requirements), and they were therefore well received by the architects. One of the advantages of the wikis is their flexibility, i.e., wikis are not associated to a particular modeling language or with particular formats of documents. They are flexible enough to allow any architect role to pass on any type of information she/he thinks is relevant to another architect or stakeholder. Another advantage of wikis is that they are lightweight and versatile platforms that better adapt to the technical and non-technical activities architects perform [22]. On the other hand, because they are so generic, wikis provide limited support for the interdependencies among architecture roles. Some of the studied companies, e.g. Company D, use customized trackers that can partially address this gap, but this is dependent on the users' adoption.

Finally, there is another aspect that could potentially influence IT architects' activities and should be approached by tools or strategies that aim to support them: software traceability. Spanoudakis and Zisman [25] describe software traceability as "the ability to relate artifacts created during the development of a software system to describe the system from different perspectives and levels of abstraction with each other, the stakeholders that have contributed to the creation of the artifacts, and the rationale that explains the form of the artifacts". One of the many advantages of traceability is that it provides an understanding of project artifacts in reference to the context in which they were created or in reference to other artifacts. This understanding is particularly important when people who contributed to the creation of the artifacts are not the same ones who need to understand those artifacts [25], and also in information system development where architects are the "walking architectures": the main source of information in system development projects [3,4,23]. In short, traceability tools can be used to relate a system from its requirements and commercial aspects to the source code, indicating the path used and the decisions made during system development and providing support to activities of different stakeholders.

As we discussed in the beginning of this section, we were not able to find tools that support traceability aspects among the different architect roles activities. Nevertheless, even the wiki, trackers and requirements issues discussed above can be related to traceability issues and, thus, could be supported by a tool that better address traceability activities. For instance, the trackers used by Company D and others provide an initial support to

[4] Although some of the companies we studied used some the tools mentioned in the previous paragraph.

traceability activities, since they can be used to delegate activities from an architect role to another and, consequently, offer support to one architect role to track the activities developed by the previous architect. On the other hand, wikis can store the whole evolution of system artifacts, since stakeholders can store any kind of artifact (though it depends on the people to properly use it). Both trackers and wikis can help in problems related to connecting requirements management and IT architecture activities, but they provide only an initial support to it. Future tools should also use the information provided by trackers and stored in wikis to generate and store traceability items and links in a more automatized way in order to reduce the effort necessary to create traceability information [25]. Thus, a traceability tool that aggregates these characteristics considering architect roles interconnections would not only support the boundary spanner aspect of architect activities, but also improve traceability and daily work for architects.

8. Limitations

In this section we describe the limitations of our research. In quantitative studies this is traditionally done by describing aspects regarding the construct validity, external validity and internal validity of the study. However, Klein and Myers [35] argue that some criteria that are useful for evaluating research conducted according to the natural science model (positivist research) are inappropriate for judging interpretive research. We believe that is exactly what happens with construct validity, external validity and internal validity: they are criteria that are useful for judging positivist research, but not interpretive and qualitative research. Therefore, we will not report these aspects in this paper. However, "it does not follow that there are no standards at all by which interpretive research can be judged". In fact, Klein and Myers [35] present a set of seven principles that they argue might be used to *conduct* and *evaluate* interpretive field research. Below, we will describe how the research described in this paper observes these principles.

The *first* principle, the hermeneutic circle, suggests that all human understanding is achieved by iterating between considering the interdependent meaning of parts and the whole that they form. In our case, this was achieved by shifting the focus of analysis from the work performed by *one* IT architect to the overall context of the work performed by *all* the other architects. As we discussed in Section 4, the organization of the IT architects' work was not our original focus, but evolved from the data collection and analysis we conducted. The *second* principle focuses on the contextualization, i.e., the "historical, political, and economic context" in which the study was conducted. In our case, since there were nine different companies in which the data was collected, we opted for describing the context only for the companies in which this aspect was relevant. For instance, at the end of Section 5.1 we describe the context of companies F and G because they have similarities. The Principle of Interaction Between the Researcher and the Subjects is the *third* one and requires critical reflection on how the research materials were socially constructed through the interaction between the researchers and participants [35]. This means, for instance, that it is important to document "the ways in which data collection and interpretation activities affected each other" (*ibid.*). We believe this aspect was described in rich details in Section 4.3 where we described each iteration of data collection and analysis. The *following* principle, abstraction and generalization, argues that interpretive researchers should "generalize their findings to theoretical constructions of interest". In this paper this is done through the usage of Carlile's 3T's [29] framework for knowledge management. By using this framework, we were able to propose a theory of how IT (and software) architects perform their daily work.[5] In addition, when we discuss our results in the context of previous research (Section 6), we find additional evidence supporting our theory.

The *fifth* principle, the Principle of Dialogical Reasoning requires sensitivity to possible contradictions between the theoretical preconceptions guiding the research design and actual findings ("the story which the data tell") with subsequent cycles of revision. Its most fundamental point is that "the researcher should make the historical intellectual basis of the research as transparent as possible to the reader". As we described in Section 4.3, in our study we initially began studying the work of software architects in distributed projects, assuming that the work of such architects would be different from software architects working in collocated projects. As we collected data we changed the focus to software architects in general, and then to IT architects. In other words, we described our original focus on distributed software development and how this focus changed over time. The *sixth* principle, the Principle of Multiple Interpretations, requires sensitivity to possible differences in interpretations among the participants as are typically expressed in multiple narratives or stories of the same sequence of events under study. In other words, this principle emphasizes the importance of documenting the views of other interest groups. We addressed this aspect by interviewing professionals who were not formally employed as IT architects, but who performed architectural related activities. These professionals also worked in different companies. Moreover, we reviewed the results from the interviews, comparing with the partial research results. Finally, *the suspicion principle* recommends that "authors adopt a critical perspective and do not take their informants' views at face value" (*ibid.*). In this case, as we described in our methodology section, we argue that we addressed this concern using theoretical sampling [5], i.e., by choosing informants that could either extend or question our theory in each new data collection period.

Furthermore, as we described in Section 4.3.4 we performed *member checking*, i.e., we validated our theory by presenting it both to employees of different organizations and to new employees of organizations in each of whom we had already collected data to verify whether our theory would apply to them. Interviews were audio-taped and transcribed and we documented our analysis process so as to create an *audit trail*. These two mechanisms, member checking and audit trail, are two important mechanisms used to minimize threats to validity in qualitative research [36].

Finally, it is important to mention that due to constraints of time and resources, we did *not* reach theoretical saturation [5], which means that we cannot argue that our theory is fully "developed in terms of density and variety". In other words, our theory might vary under some circumstances. For instance, organizations might adopt different types of architects who perform different roles. Despite that, we are still confident in our results for several reasons. First of all, we have described our concern and approaches to guarantee the validity of our theory and results in this section. Second, as discussed before, the results of our theory are to some extent validated by previous related work. And, finally, our theory explains what we observed in our data, allowing us to perform a theoretical generalization. We believe this might provide an important starting point for further research.

9. Conclusions and future work

We studied IT architects aiming to understand how they performed their work in practice. In order to do so, we adopted a

[5] In other words, Klein and Myers [35] argue that interpretive qualitative research should aim for providing *theoretical generalization* instead of *statistical generalization*.

qualitative approach based on semi-structured interviews and grounded theory. We conducted twenty-seven interviews with twenty-two informants from nine different companies divided into four iterations of data collection and analysis. Each iteration of data collection and analysis refined our understanding of their work, allowing us to conduct new interviews that could extend or question our results. At the end of the process, we validated our results with IT architects who either work for the companies studied or work in different companies.

In our interviews, aligned with IASA's suggestion, we noticed that organizations divide the IT architecture activities among different architect roles. Each role performs specific activities and engages in communication and coordination with specific stakeholders. IT architects also heavily depend on one another to perform their work. In addition, they act as boundary spanners in the organization, each one dealing with stakeholders from one domain and providing information for another role in a different domain. To be more specific, IT architects transfer, translate and transform [29] knowledge across different boundaries facilitating the diffusion of knowledge about the software system being built. Our research highlights the importance of the interconnections among architect roles and, as a consequence, the limited support for these interconnections provided by current tools. Overall, we argue that any approach (tool, framework, or modeling language) that aims to properly support architecture activities also needs to provide support for the interconnections and collaborations among the different architect roles. Currently, most architecture tools tend to address these roles separately, and usually do not support all different roles.

Although we did not find one same person playing more than one role in the same project, this is possible. We only found the case of a person playing her/his role and performing some activities of another role (as it happens in companies in the group called "Partially defined roles"), but not all of them. Further research is needed to better explore this scenario and to analyze its implications for collaboration and information flow.

Finally, it is important to mention that in some companies an enterprise architect is not a formal role, but since the activities of this type of architect still need to be performed, they are performed by the solution architect alongside with his/her own activities. This is an interesting aspect because it means that less collaboration is necessary because only one person is doing the work. On the other hand, it might mean that this same person might be overwhelmed by the amount of work she/he needs to perform. This is a particularly interesting because it might affect the design of tools to support IT architects.

Acknowledgments

This research received funding from CNPq through the "Edital Universal 2008" Process Number 473220/2008-3, FAPESPA through "Edital Universal" number 003/2008 and Edital 014/2008 (project "Rede Paraense de Pesquisa em Tecnologias de Informação e Comunicações"), from CAPES through a M.Sc. scholarship granted to the first author, and from the PDTI program, financed by Dell Computers of Brazil Ltd. (Law 8.248/91). The fifth author was also supported by CNPq (483125/2010-5 and 560037/2010-4).

Appendix A. Interview guides

A.1. First data collection interview guide

–**General questions**

(1) What is your full name?
(2) How long have you worked in this company?
(3) What is your job title in the company?
(4) How long have you held this position?
(5) What are the most important projects you are currently involved?
(6) What are your activities as <position>?
(7) How many people work with you? What is their work?

–**About the project and distributed software development**

(8) Do any of the projects you mentioned involve people in different locations, i.e., distributed development? Which one (if more than one request that he/she describes an example)?
  a. How many different teams work on this project?
  b. Where are the teams located?
  c. How is the task division?
  d. What part of the project is your team's responsibility?
  e. How is your team's part related with the other team parts?
(9) If the interviewee is not currently involved in any Project with DSD: when was the last time you worked in a distributed Project?
  a. (As an architect?
(10) How was the work divided among the teams?
  a. Who did the division?
When?
  b. Have these teams worked together before?

–**About the interviewee work**

(11) What are your specific activities in this project?
(12) Was an architecture defined to be used in this project?
(13) What was the process used to define the architecture in this project?
  a. What was your part in this stage?
  b. How many people participate in this task? Who are the other people? Where are they located?
  c. How was the team that defined the architecture formed? How are the people in this team chosen?
    i. Do non-architects participate in this stage?
    ii. Members of different teams? Customers? Experts? Developers? Indications?
iii. Who contacts these people?
iv. What is the role of each member of the architecture definition team?
(14) When and how was the division of work among teams defined in this project?
  a. Was it you who assigned the division of the modules?
  b. Were the modules divided among the teams following any special criteria? What was it?
  c. How was it identified which team was the most appropriate for developing each module?

(*continued on next page*)

d. Did the work division consider the architecture? How?
e. Did the work division consider the DSD? How?
f. Did the work division consider the company structure? How?
(15) After the architecture definition, what was the next step? For example, who:
a. Presents the architecture for the rest of the teams?
b. Negotiates the architecture?
c. Receives feedback from the others involved?
d. Modifies the architecture based on the feedback?
(16) How was/is the schedule in this project made? Considering the distribution (DSD)? How?

–About the architecture

(17) What factors influence in the architecture/module definition?
(18) Do you think software architecture is a critical success factor in a (collocated or distributed) project? Why?
(19) What are the main difficulties related to the architecture found by the team?
(20) At which development stage were these difficulties identified?
(21) Does team distribution affect the architecture?
a. If one team is expert in a specific aspect, are all features related to this activity aggregated in one module?
(22) Can you describe some strategies used to distribute the components development in this project?
a. Are complex components divided? Or are they developed completely by one team?
b. Are features aggregated in components according to teams' expertise? Or are the components defined and only then is the distribution defined?
(23) If the interviewee answered yes to question 14e: Can you give an example of strategy used in the architecture to deal with development distribution? Something that could be developed in another way, but in DSD you needed to think of a different approach?
(24) Is there any strategy that was successful in other projects and now is often reused in distributed projects?

–About the interaction among teams

(25) How is the dependence among the work of the different teams?
(26) How are the dependencies managed in the project?
a. Are the interfaces defined?
b. When and how is the module integration performed?
c. If a problem occurs when integrating the modules, how it is solved? Has this happened before?
(27) How often do you need to interact with people from other teams in this project (located elsewhere)?
(28) How does this interaction happens?
(29) How often do the teams need to Interact with each other?
(30) Do you contribute to the interaction among the teams? How?
(31) How is the development information of one team passed to the others?
a. Meetings? E-mails? Reports?
(32) If the structure of one team's component needs to be changed, how does that affect the other teams? How is this change communicated to the other teams?

–About the company

(33) What are the target markets of the software developed by the company?
(34) What are the platforms / programming languages used by the company?
(35) Generally speaking, at what stage of the software development cycle does the company usually work on the software architecture?
(36) Do you think software architecture is a success critical factor in a distributed project? Why?
(37) What difficulties, related to the software architecture, can be found by teams that work with DSD in opposition to collocated teams?
(38) How can these difficulties be minimized?
(39) Do you think the kind of platform/programming language or product/service developed can influence in the success of a distributed project? Which ones?

A.2. Second data collection interview guide

–General questions

(1) What is your full name?
(2) How long have you worked in this company?
(3) What is your job title in the company?
(4) How long have you held this position?
(5) What are the most important projects you are currently involved in?
(6) What are your activities as <position>?
(7) How many people work with you? What is their work?

–About the project and distributed software development

(8) Do any of the projects you mentioned involve people in different locations, i.e., distributed development? Which one (if more than one request that he/she describes an example)?
a. How many different teams work in this project?
b. Where are the teams located?
c. How is the task division?
d. What part of the project is your team responsibility?
e. How is your team's part related with the other team parts?
(9) If the interviewee is not currently involved in any Project with DSD: when was the last time you worked in distributed Project?
a. As an architect?
(10) How was the work divided among the teams?
a. Who did the division?
b. When?
c. Have these teams worked together before?

–About the interviewee work

(11) What are your specific activities in this project?
(12) What was the "process" used to define the architecture in this project?
a. What was your part in this stage?
b. How many people participate in this task? Who are the other people? Where are they located?
c. How was the team that defined the architecture formed? How are the people in this team chosen?
i. Do non-architects participate in this stage?
ii. Members of different teams? Customers? Experts? Developers? Indications?
iii. What is the role of each member of the architecture definition team?
(13) When and how was the division of work among teams defined in this project?

a. Did the work division consider the architecture? How?
b. Did the work division consider the DSD? How?
c. Did the work division consider the company structure? How?

–**About the architect work**
(14) What do you think an architect needs to know or how does this person need to act in order to be considered a good architect?
(15) How many architects usually work in a project in your company?
a. Do they have different profiles (interests, etc.)? What are the specificities of these profiles?
b. What are the benefits of this configuration?
c. If answered yes to a), what are the tasks of each type of architect?
d. If answered yes to a), what is your profile? What is your work and how it is related to the work of the other profiles?
e. If answered yes to a), where are the architects located?
i. All in the same location or distributed among the teams?
ii. Any special reason for this configuration?
(16) How is the interaction among the architects?
a. Do they work together?
b. How do they interact during the architecture definition?
c. How do they interact with other stakeholders?
(17) How is the participation of the architect in the communication among the teams of different sites? And in the communication inside the team(s) of his/her site?
(18) Do you often communicate with other non-developers stakeholders?
a. How often?
b. What are the reasons?
c. Only from your site or from the others too?
(19) Do other stakeholders (developers, customers, managers, etc.) come to you to ask questions about the architecture?
a. How often?
b. Only from your site or from the other too?
c. Are there others who can also answer these questions? Who are they and what are their roles in the project?

–**About the architecture**
(20) How is the architecture represented?
a. Text? Source code? Boxes and arrows? Classes, packages and diagrams?
b. Different visions?
(21) What is the detail level of the architecture representation?
(22) How is this architecture representation used?
a. Design?
b. Communication among developers?
c. Change communication?
d. Communication to develop new features?
e. Communication for bug fixing?
f. Implementation feedback?
g. Work and responsibilities distribution?
(23) How is the architecture representation updated?
a. Never? Regularly controlled? Continuously? Related to the general or releases plan? Only when problems occur?
(24) What factors influence in the architecture/module definition?
(25) Do you think software architecture is a critical success factor in a (collocated or distributed) project? Why?
(26) What are the main difficulties related to the architecture found by the team?
(27) At which development stage were these difficulties identified?
(28) Does team distribution affect the architecture?

–**About the interaction among teams**
(29) How is the dependence among the work of the different teams?
(30) How are the dependencies managed in the project?
a. Are the interfaces defined?
b. When and how is module integration performed?
c. If a problem occurs when integrating the modules, how it is solved? Has this happened before?
(31) How often do you need to interact with people of other teams in this project (located elsewhere)?
(32) How does this interaction happen?
(33) How often do the teams need to Interact with each other?
(34) Do you contribute to the interaction among the teams? How?
(35) How is the development information of one team passed to the others?
a. Meetings? E-mails? Reports?
(36) If the structure of one team's component needs to be changed, how does it affect the other teams? How is this change communicated to the other teams?

A.3. Third and fourth data collection interview guide

–**General questions**
(1) What is your full name?
(2) How long have you worked in this company?
(3) What is your job title in the company?
(4) How long have you held this position?
(5) What are the most important projects you are currently involved in?
(6) What are your activities as <position>?

–**About the architect work and the architecture?**
(7) What are your specific activities in this project?
(8) How was the architecture defined?
a. What was your part in this stage?
b. Who defines the architecture?
i. Do non-architects participate in this stage?
ii. Members of different teams? Customers? Experts? Developers? Indications?
iii. What is the role of each member of the architecture definition team?
(9) What factors influence in the architecture/modules definition?
(10) How is the architecture represented?
(11) What do you think an architect needs to know or how to act to be considered a good architect?
(12) Which roles interact more often with the architect?
(13) How many architects usually work in a project in your company?
a. Do they have different roles? What are the specificities of these profiles?
b. What are the benefits of this configuration?
c. If answered yes to a), what are the tasks of each type of architect?

(*continued on next page*)

d. If answered yes to a), what is your profile? What is your work and how it is related to the work of the other profiles?

(14) How is the interaction among architects and the roles of question 12?
a. Do they work together?
b. Do they interact during the architecture definition?
c. How do they interact with other stakeholders?

(15) How is the participation of the architect in the communication among the teams?

(16) Do you often communicate with other non-developers stakeholders?
a. How often?
b. What are the reasons?

(17) Do other stakeholders (developers, customers, managers, etc.) come to you to ask questions about the architecture?
a. How often?
b. Are there others who can also answer these questions? Who are they and what their role in the project?

–About the transformation process

(18) How is the project development process from the customer request to the developers?
a. Which artifacts are developed?
b. What is the content of these artifacts?

(19) Explain the transformation theory and ask if something like that happens in the interviewee's company.
a. How is the division of activities?
b. Does the architect help in disseminating information?
c. What are the advantages of this configuration?
d. Any more information to add?

–About difficulties/problems/bad practices

(20) What are the main difficulties related to the architecture found by the team?

(21) Have you ever faced a situation where requirements were specified and implemented in accordance with the specification but it was discovered that they were not what the customer wanted?
a. Why does that happen?
b. How do you solve it? What do you think would help?
c. What do you expect from the analyst in this aspect?
d. Do you know the training the analysts have? Requirements engineering?

(22) Have you ever faced a situation where you developed a good architecture, which meets the needs, works, is elegant, but becomes too difficult or complicated for the team to develop?
a. What is done when that happens?
b. How do you avoid it?

(23) Bad practices?

References

[1] IASA. <http://www.iasaglobal.org/iasa/default.asp>.
[2] The Open Group. <http://www.opengroup.org/>.
[3] K. Smolander, Four metaphors of architecture in software organizations: finding out the meaning of architecture in practice, in: Proceedings of the First International Symposium in Empirical Software Engineering, Nara, Japan, IEEE Press, 2002.
[4] R.E. Grinter, System architecture: product designing and social engineering, in: Work Activities Coordination and Collaboration, San Francisco, CA, USA, ACM Press, 1999.
[5] A. Strauss, J. Corbin, Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory, SAGE publications, Thousand Oaks, CA, 1998.
[6] J. Wilt, IASA's Five Pillars of IT Architecture. Microsoft tech. ed., 2010. <http://www.msteched.com/2010/Europe/ARC204> (access 16.11.10).
[7] The Open Group. Business Executive's Guide to IT Architecture Have You Thought, 2004. <http://www.opengroup.org/bookstore/catalog/w043.htm> (access 15.11.10).
[8] The Open Group, Information Technology Architect Certification Program – Conformance Requirements (Multi-level), England, 2008. <http://www.opengroup.org/itac> (access 15.11.10).
[9] D. Akenine, A study of architect roles by IASA Sweden, Arch. J. – Role Arch. 15 (2008) 22–25.
[10] J. Hofstader, We don't need no architects!, Arch J. – Role Arch. J. 15 (2008) 2–6.
[11] A. Unde, Becoming an architect in a system integrator, Arch. J. – Role Arch. 15 (2008) 7–9.
[13] Zachman J. 2008. Zachman Framework. Available in: <http://zachmaninternational.com/2/Zachman_Framework.asp>. Access in February 26, 2011.
[14] M. Penker, H. Eriksson, Business Modeling with UML: Business Patterns at Work, John Wiley & Sons, Inc., New York, NY, USA, 2000.
[15] Object Management Group, Object Management Group/Business Process Management Initiative, 2011. <http://www.bpmn.org/> (access 27.02.11).
[16] J. Zachman, The Zachman Framework: A Primer for Enterprise Engineering and Manufacturing, 2003. <http://www.zachmaninternational.com> (access 26.02.11).
[17] The Open Group, TOGAF Version 9 – The Open Group Architecture Framework (TOGAF), 2009. <http://www.opengroup.org/togaf/> (access 20.02.11).
[18] J. Schekkerman, Enterprise Architecture Tool Selection Guide, 2009. <http://www.enterprise-architecture.info/EA_Tools.htm> (access 20.02.11).
[19] Verbi Software, MAXQDA – The Art of Text Analysis. <http://www.maxqda.com/> (access 05.01.10).
[22] M. Ali Babar, R. de Boer, T. Dingsøyr, R. Farenhorst, Architectural knowledge management strategies: approaches in research and industry, in: Workshop on Sharing and Reusing architectural Knowledge, SHARK'07, 2007.
[23] H. Unphon, Y. Dittrich, Software architecture awareness in long-term software product evolution, J. Syst. Softw. 83 (2010) (2010) 2211–2226.
[24] D.L. Jorgensen, Participant Observation: A Methodology for Human Studies, SAGE publications, Thousand Oaks, CA, 1989.
[25] G. Spanoudakis, A. Zisman, Software traceability: a roadmap. handbook of software engineering and knowledge engineering, in: S.K. Chang (Ed.), Recent Advancements, vol. III, World Scientific Publishing Co., 2005. ISBN 981-256-273-7.
[26] P. Carlile, E. Rebentisch, Into the black box: the knowledge transformation cycle, Manage. Sci. 49 (9) (2003) 1180–1195.
[27] B. Curtis, H. Krasner, N. Iscoe, A field study of the software design process for large systems, Commun. ACM 31 (11) (1988).
[28] R. Hiranpruk, Introduction to IASA, 2006.
[29] P. Carlile, Transferring translating and transforming: an integrative and relational approach to sharing and assessing knowledge across boundaries, Org. Sci. (2003).
[30] M.C. Figueiredo, C.R.B. De Souza, M.Z. Pereira, J. Audy, R. Prikladnicki, On the role of information technology systems architects, in: 18th Americas Conference on Information Systems, Seattle, 2012.
[31] C. Seaman, Qualitative methods, in: Guide to Advanced Empirical Software Engineering, Springer-Verlag, Londres, 2008, pp. 35–62.
[32] J.D. Herbsleb, D. Moitra, Global software development, IEEE Softw. 18 (N2) (2001) 16–20.
[33] K. Smolander, M. Rossi, S. Purao, Software architectures: blueprint, literature, language or decision?, Eur J. Inform. Syst. 17 (6) (2008) 575–588.
[34] D. Ameller, C. Ayala, J. Cabot, X. Franch, How do software architects consider non-functional requirements: an exploratory study, in: 20th IEEE International Conference on Requirements Engineering (RE), 24–28 Sept. 2012, pp. 41, 50.
[35] H.K. Klein, M.D. Myers, A set of principles for conducting and evaluating interpretive field studies in information systems, MIS Quart. 23 (1) (1999) 67–94.
[36] M.Q. Patton, Qualitative Research and Evaluation Methods, Sage Publications, Newbury Park, CA, 1990.
[37] L.T. Lopes, J.L.N. Audy, A requirements engineering process model for distributed software development – lessons learned, in: International Conference on Enterprise Information Systems, 2008, pp. 117–122.
[38] F. Calefato, D. Damian, F. Lanubile, Computer-mediated communication to support distributed requirements elicitations and negotiations tasks, Emp. Softw. Eng. 17 (6) (2012) 640–674.
[39] R. Prikladnicki, E. Carmel, Is time-zone proximity an advantage for software development? The case of the Brazilian IT industry, in: Proceedings of the International Conference on Software Engineering, ACM/IEEE, San Francisco, USA, 2013.