Architecture Alignment in a Large Government Organization: A Case Study

R.J. Wieringa, P.A.T. van Eck, and H.M. Blanken

Center for Telematics and Information Technology, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands
(roelw|vaneck|blanken)@cs.utwente.nl

Abstract. In this paper we view IT architecture as the structures present in the entire information technology support used by an organization. We report on a detailed case study of an operational IT architecture process, in which we investigated the relationship between IT architecture and business context. We analyze this process in terms of a conceptual framework for IT architecture presented in earlier work. The major findings are that application architecture is designed by aligning applications to the business process structure; and that IT infrastructure architecture is designed by aligning it to technological trends rather than to business goals and problems. This paper reports about a single case study done in a series of case studies. We discuss the generalizability of the findings from this case study, and discuss topics for further research.

1 Introduction

Architecture alignment is the mutual adjustment of IT architecture and business architecture. For large information-processing organizations such as banks, insurance companies and government organizations, continuous architecture alignment is crucial to survive. The IT departments of these organizations are faced with frequent mergers, which requires them to integrate several IT structures that are pairwise incompatible, and with frequent reorganizations, which necessitates adjustment of the IT structures to new organization structures. In this context IT architecture design is a complex problem that is never solved but must continually be attented to.

Our research can be positioned between Software Engineering and Business Science. Most current software engineering research [1–3] does not provide all the techniques to deal with architecture alignment, because it focusses on the internal structure of one software system and does not include in its scope the business context of an ensemble of software systems that together realize all information processing needs of an organization. On the other hand, business scientists who study architecture alignment do not come up with operational design guidelinest [4–6]. Opdahl makes a start with defining guidelines for representing application alignment [7]. However, typical business studies investigate the alignment of business objectives and IT objectives, or management issues such as e.g. the conditions of IT personnel management. In short, the practicing

IT architect misses operational guidelines for relating IT architecture to business architecture.

The GRAAL project aims to identify operational guidelines for business-IT architecture alignment for large information-processing organizations. We reach this goal by doing case study research in these organizations. To be able to compare the analysis results in different organizations (and sometimes even in different parts of the same organization), we developed a conceptual framework for business and IT architectures, which we reported on earlier [8]. In the current paper we report on a case study of architecture alignment performed at a large government organization that we will call BIGIT (Big Government IT organization). BIGIT has about 37000 users spread out over different locations in The Netherlands. BIGIT is part of a large government organization that we will call BIG. BIGIT provides IT development and maintenance services for BIG. BIG is partitioned into a number of departments, one of which we will simply call D. We will focus our case study on the services provided by BIGIT for D.

We should state at the outset that we report on *case study research* [9]. This means that we have observed what happened in an IT department and analyzed our observations. Case study research should not be confused with experimental research. The goal of experimental research is to make inferences about a *population*. This is usually done by taking a sufficiently large random selection of cases from a population and generalize to the whole population by means of statistical methods. Case study research does not generalize to a population, but to *theory* [9]: every extra case study performed in which patterns predicted by a theory are found increases the extent to which we believe the theory is true. Case study research is also different from action research. In action research, a researcher actively tries to influence the outcome of the case at hand by performing actions, and evaluating whether the actions had the effects predicted by theory. In case study research, the case is left undisturbed. This paper reports on an exploratory case study, which is a case study that is not carried out starting from a set of hypotheses, but aims to find hypotheses to be validated in further case studies.

2 The GRAAL Framework

In order to analyse cases, we need a conceptual framework and vocabulary in terms of which the cases are described. An analysis of a number of frameworks for information systems and software development, industrial product development and systems engineering led to a simple framework for software and business shown in Fig. 1 [10, 11].

The core of the framework consists of a stack of layers. Each layer consists of entities (hardware, software, people, organizational units, customers). Entities at lower layers provide services to entities at higher layers. At BIGIT, we distinguish, from the bottom up, the layers shown in Fig. 1. The two lowest layers need explanation:

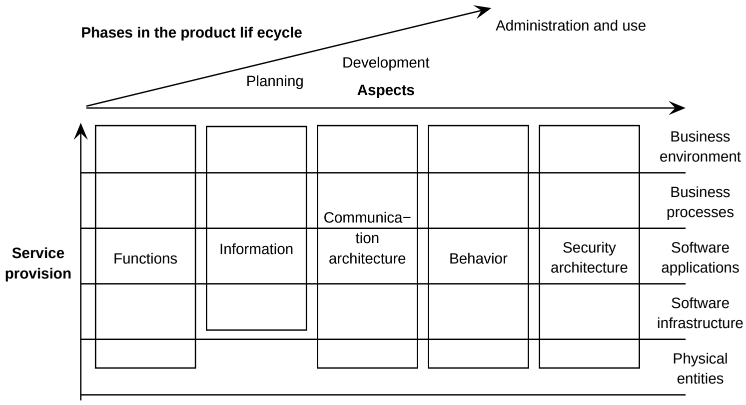

Fig. 1. The GRAAL Framework. An additional dimension of our framework, not shown, concerns the refinement level at which we describe entities at each layer.

– Physical entities are, for example, computers, networks, buildings, furniture, etc. An entity is physical if it can be described using physical dimensions such as meters, kilograms, amperes, and seconds.
– The software infrastructure consists of operating systems, network software, middleware, DBMSs. A software entity is part of the infrastructure if it provides services of general use, i.e. not containing specific knowledge of the organization in which it functions.

The entities at each layer have properties, which we can classify under *aspects.* Each entity has a *function* for its environment. (In this paper, we use the word "function" as synonym of "service".) All entities above the physical layer have an *information* aspect. For example, software manipulates symbols with a well-defined meaning, business processes manipulate data, and customers supply data to BIGIT. All entities *communicate* with other entities, they all have *behavior* over time, and in BIGIT, *security* is important at all layers. In other organizations, other aspects may be important too, such as reliability of interoperability.

Each entity has a *life cycle* that starts with a planning stage and ends with the management of the entity in use. This is complicated by the fact that many entities may exist in different versions, and each of these versions goes through a planning, development and deployment stage. At any point in time, several versions of an entity may exist in different stages of its life cycle.

Finally, a dimension not shown in the diagram concerns the *description* of entities. Each entity can be described by a combination of text and diagrams. These descriptions can be very abstract (little detail) or very refined (a lot of detail). Comparisons have been published elsewhere [11, 8].

3 The Case

We will focus our case study on the services provided by BIGIT for one of the departments of BIG, called D. We have analyzed ten architecture documents of the IT systems of D, developed and maintained by BIGIT (Fig. 2). Documents 1 through 5 describe applications and information systems at various levels of detail. Documents 6 and 8 treats *infrastructure domains* in BIGIT. Finally, documents 7, 9 and 10 deal with various aspects of the development process of software systems. Based on these documents, we drafted an initial analysis of business-IT alignment at D and verified this with the IT architects of BIGIT working for D. After adjusting the analysis to feedback that we received, we presented the final result to the BIGIT architects. Here, we present this final result in anonymized form.

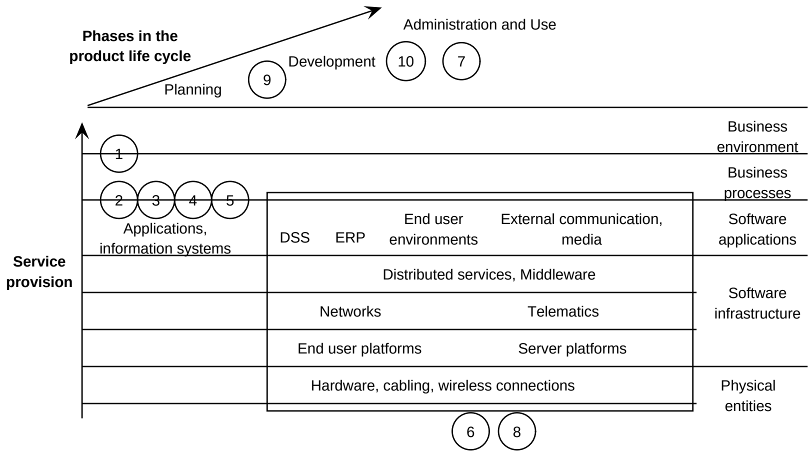

Fig. 2. Analyzed documents. The horizontal dimension of Fig. 1 is not used.

4 Findings

4.1 Applications and infrastructure

The first finding in our case study is that there is a clear separation between applications and infrastructure in terms of acquisition and maintenance. We adapted our framework to this by distinguishing these two layers in Fig. 1. A software system is an *application* if it provides services for a specific user group in BIG and contains knowledge that is specific for this user group. Applications usually play a role in a particular business process. They are acquired (built or bought) specifically for their user groups and for particular business processes. Failures are felt by this user group and in this business process only.

A software system is part of the *infrastructure* if it provides services for all users and does not contain any knowledge about its particular context. Infrastructure is not acquired for any particular user group or any specific business process. Like a central heating system, infrastructure is expected to be available wherever and whenever needed. Failures are felt in the entire user population.

Figure 2 shows that infrastructure at BIGIT is a complex consisting of several layers, ranging from hardware and physical connection technology, to operating systems middleware, telecom services, decision support systems, ERP systems and office software available on all user stations. Not only the top layer of infrastructure is directly accessible to users. Users directly interact with hardware, operating systems and communication technology and in one way or another depend on all infrastructure elements. This puts a high premium on the quality of service of an infrastructure.

Application development is event-driven. Usually, there is a change request from a user group, or a management directive to implement a new service. The application development process then generates a steady flow of documents, starting from the change request itself, to an initial requirements definition, top-level architecture design and so on to the finally delivered documents: Source code, executables, technical and user documentation.

Infrastructure development is time-triggered. It takes place periodically, for example yearly, and produces less documents. These documents are not oriented towards a service to be delivered towards users but towards business goals to be satisfied and the available technology to satisfy them. We return to this below.

Given this difference between applications and infrastructure, we can expect application architecture to be structured according to user groups and their business processes. Infrastructure is not oriented to any particular user group, and its architecture is structured according to technological domains. There is thus a tension in architecture alignment between applications and infrastructure. This will be confirmed in the next two sections.

4.2 Application Alignment Relations

Application alignment at BIGIT takes place in three steps, which produce three documents. Starting from a business strategy, a design of the business processes is produced. This is the basis for a document describing the architecture of the entire application layer, which in turn is the basis for the architecture of each individual application. The actual process is not as rational as suggested here, but we will see that it does not depart very much from this rational process.

At BIGIT, the application layer is split in two, an information system layer and what we will call a “pure” application layer. The information system layer contains the data storage systems and the pure application layer contains the data manipulation systems. Pure applications are stateless; all state is maintained by the information systems. The term “application” is thus ambiguous because it can refer to pure, stateless applications only, and to information systems plus stateless applications jointly. This ambiguity turns out not to be harmful and we will provide disambiguation where needed.

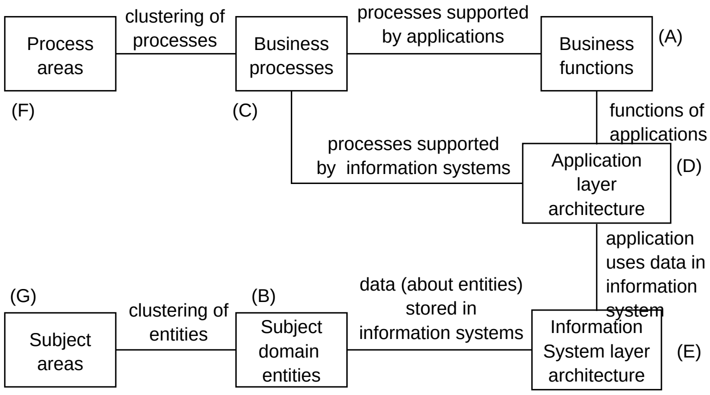

Fig. 3. Application alignment relations.

Fig. 3 shows the application layer architecture descriptions produced at BIGIT for their customer D. We only explain the major descriptions.

– (A): A list of *business functions* of D is made. For each function, its contribution to the business goal of D is described. Functions are organized into a function refinement tree with the business mission of D at its root and elementary functions at its leaves.
– (B): A description of *subject domain entities* of D is made. The subject domain of a business is the part of the world about which the business contains data. (Remember that D is an information-processing organization.) The subject domain description is cross-checked with the business function descriptions by checking whether the entities about which a function needs data, are described in the subject domain description. (This cross-check is not represented in Fig. 3.)
– (C): A description of end-to-end *business processes* of D is made, leading from customer of D to customer of D. This is cross-checked with the function refinement tree (does every process support a function and vice versa?).
– (D): The descriptions of functions and processes are used to identify an *application layer architecture*. This is a list of applications and for each application, its role in one or more business processes, and its contribution to business functions.
– (E): The application layer architecture and subject domain description are used to design an *information system architecture*. This is a list of information systems and for each information system, the data it stores and the applications it supports.

These descriptions have been produced using a method called Panfox [12], which is itself based on Information Engineering [13]. Observe that application alignment is defined at BIGIT for the functional aspects of our GRAAL framework:

– functions (business functions in Fig. 3),

– information (subject domain entities, subject areas),
– communication (the application layer and information system layer architectures, process areas),
– and behavior (business processes and process areas (F) and (G) in Fig. 3).

We observed some problems in implementing the application alignment scheme of Fig. 3, which are presented in a report [14]. Finally, for each individual application of D, there is a systematic process at BIGIT to define its internal architecture based on the application layer architecture. Basically, the application layer architecture provides the context diagram for each individual application and information system. Application architecture design is software architecture design [1–3] and due to space restrictions we will not discuss this here.

4.3 Infrastructure Design Arguments

BIGIT's customers are all other departments of BIG, of which D is only one. The infrastructure architecture is motivated in terms of four kinds of phenomena that do not refer to the needs of D but of BIG as a whole:

– *Business goals* of BIG. Examples goals are improvement of accessibility of BIG's services, improvement of customer-friendliness and facilitating mobility of BIG's employees.
– *Problems* experienced at BIG. Example problems are the high cost of maintenance, the large number of systems for which users must remember passwords, and the lack of OS/390 experts.
– *Current systems* at BIG. These range from ERP systems used to the range of operating systems in use (and in various versions), all classified according to the infrastructure domains shown in Fig. 2.
– *The current technology market trends* for off-the-shelf applications, office software, document management software, middleware, telematics software, etc., again organized according to technology domain.

These considerations transcend not only the individual business processes at D. They also transcend D. In fact, D must define a business strategy that agrees with BIG's strategy. Infrastructure alignment starts from a BIG business strategy, which covers all aspects of the business, including some very general IT aspects. This is then specialized into a strategy for the IT infrastructure only. This is in turn refined into a description of the next version of the infrastructure architecture for BIG, and the acquisitions of IT that follow from this. We observed a number of related alignment failures at BIGIT, that can all be traced to the following phenomenon: Each infrastructure domain is a technical knowledge area that requires several years of study followed by constant attention to the trade press to understand. So for each domain, there is a technical specialist gathering domain knowledge. At BIGIT, this caused a number of alignment failures:

– *Technology orientation.* Infrastructure architects follow their part of the technology market daily. This highly technical orientation makes them less sensitive to business strategies and business problems.

– *Domain islands.* The highly specialist nature of infrastructure domain knowledge even tends to isolate domain specialists from each other. The infrastructure architecture document (document 8 in Fig. 2) has one chapter for each domain. These chapters do not refer to each other and are even written using different typefonts and different diagram conventions, which indicates their lack of alignment.
– *No traceability.* Although business goals and business problems are listed extensively, none of the infrastructure design decisions were related to business goals. There was no inventory of goals expected to be satisfied by the decisions made, nor of problems expected to be solved by decisions made.
– *Technology-driven decisions.* Infrastructure design decisions were made in terms of market developments. The general trend was to add more technology to existing technology. The desire to terminating a line of technology that was more than 20 years old was included in the IT goals but was not followed by an actual decision to do so.

These consequences are all related to the highly specialist technical nature of infrastructure domains, but they do not follow necessarily from it. It is possible to make the link between a technology decision on the one hand, and business goals and problems on the other, explicit. The infrastructure architecture document (nr. 8 in Fig. 2) contains several of these links, but they are buried in a mass of details about goals, problems, legacy systems and market trends.

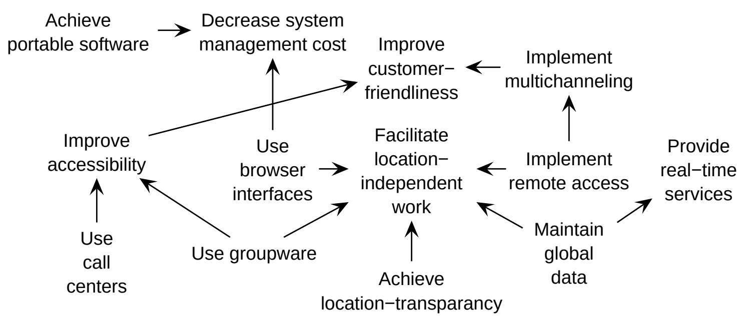

Fig. 4. Goal graph. Each node represents a goal chosen by BIG. $A \rightarrow B$ means that A contributes to B, in other words that B is a reason to choose A.

Fig. 4 shows a fragment of a goal-directed argument. This is a rational reconsruction of part of the reasoning in the infrastructure documentation. Some of the goals are ill-defined, and others are redundant.

Fig. 5 shows a problem analysis that links observable phenomena to a relevant norm. Two underlying causes lead, through a chain of causation, to the phenomenon that infrastructure management costs increase. This is a phenomenon that only becomes a problem because it conflicts with a norm, namely that

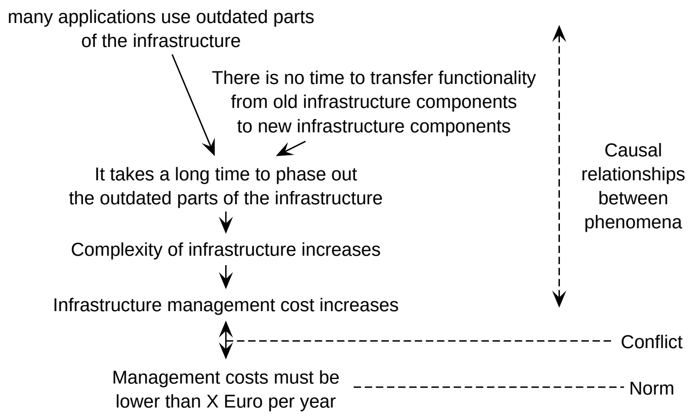

Fig. 5. problem graph.

system management costs must be below a certain amount. From this analysis one can derive a goal such as that outdated parts of the infrastructure must be replaced.

Analyses of goals and problems do occur in the infrastructure documentation, but they are sometimes implicit, and in all cases they are hidden behind extensive discussions of technology market trends. We hypothesize that this is the main reason for misalignment of infrastructure to applications. Further case study research is needed to corroborate or refine this hypothesis.

5 Discussion and Conclusions

We can derive a number of hypotheses from this case analysis.

- Application alignment takes place in a different way from infrastructure alignment, and these two can lead to a misalignment of application architecture and infrastructure architecture (sections 4.2 and 4.3)
- Applications alignment to business processes is dealt with by classical methods such as Information Engineering (section 4.2).
- For an architecture to be effective, support from user management as well as from all application architects is needed (section 4.2).
- If there is no strong management guidance infrastructure aligns to technology domains rather than to business goals and problems (section 4.3).

These hypotheses have been corroborated in a second case study and are currently being refined in a third.

We also identified a number of topics for further research:

- We need to find operational guidelines for function, process and data modeling (section 4.2). This is the domain of information system development

methods. Our case studies, not only the one at BIGIT, show that these methods give a lot of notations and process steps but hardly any operational guidelines about how to actually make modeling decisions. We made a start identifying these guidelines [10, 8] but more work needs to be done.
- Similar remarks can be made about guidelines to define application architectures (section 4.2).
- On the organization side, we also need to identify guidelines for the incorporation of legacy systems and the definition of ownership and reward structures that are critical success factors for application alignment (section 4.2).
- We merely scratched the surface of infrastructure architecture design. More work needs to bne done to identify guidelines for this (section 4.3).

Acknowledgements

Thanks are due to the IT architects of the organization called BIGIT in this paper. This research was partially sponsored by the Telematics Institute.

References

1. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley (1998)
2. Hofmeister, C., Nord, R., Soni, D.: Applied Software Architecture. Addison-Wesley (2000)
3. Shaw, M., Garlan, D.: Software Architecture: Perspective on an Emerging Discipline. Prentice Hall (1996)
4. Chan, Y.: Why haven't we mastered alignment? the importance of the informal organizational structure. MIS Quarterly Executive **1** (2002) 97–112
5. Henderson, J., Venkatraman, N.: Strategic alignment: leveraging information technology for transforming operations. IBM Systems Journal **32** (1993) 4–16
6. Leifer, R.: Matching computer-based information systems with organizational structure. MIS Quarterly **12** (1988) 63–73
7. Opdahl, A.: Model-supported alignment of information systems architecture. In Kangas, K., ed.: Business Strategies for Information Technology Management. Idea Group Publishing (2003)
8. Wieringa, R., Blanken, H., Fokkinga, M., Grefen, P.: Aligning application architecture to the business context. In: Conference on Advanced Information System Engineering (CAiSE 03), Springer (2003) 209–225 LNCS 2681.
9. Yin, R.: Case Study research: Design and Methods. Sage Publications (2003) Third Edition.
10. Wieringa, R., Blanken, H., Fokkinga, M.: Documenting the ICT architecture of TSI. CTIT Technical Report 01-42, University of Twente (2001)
11. Wieringa, R.: A survey of structured and object-oriented software specification methods and techniques. ACM Computing Surveys **30** (1998) 459–527
12. Sanden, W.v.d., Sturm, B.: Informatiearchitectuur: De infrastructurele benadering. Panfox Holding (1997)
13. Martin, J.: Information Engineering. Prentice-Hall (1989) Three volumes.
14. Wieringa, R., van Eck, P., Blanken, H.: Architecture alignment in a large government organization: A case study. CTIT technical report, University of Twente (2004)