A research towards bridging the gap between Information System Architecture and Business Architecture based on architecture patterns

Wang, H.

Award date:
2012

Eindhoven, August, 2012

A research towards bridging the gap between Information System Architecture and Business Architecture based on architecture patterns

by Huiyu Wang

Student identity number 0758208

in partial fulfilment of the requirements for the degree of

Master of Science
in Business Information Systems

Supervisors:
dr.ir. J.J.M. Trienekens
prof.dr. R. Kusters
dr. L. Somers

TUE. Department of Mathematics and Computer Science
Series Master Theses Business Information Systems

Subject headings: Enterprise Architecture, Business Architecture, Information System Architecture

Abstract

In this report, we investigated the gap between Business Architecture and Information System Architecture. Based on literature studies, we make proposals toward bridging the gap existed. With the help of our work, business and information system architects could understand better in both kinds of architectures and how they are interrelated. We make three proposals in the latter part of this report.

Executive Summary

Nowadays, Enterprise Architecture has been mentioned in many places and used broadly in many enterprise applications. However, we found that a successful enterprise application is sometimes hard to be developed and costs lots of resources in terms of human resources and money. The time needed is often very long due to its complexity. Sometimes, people who expertize on business functions may have few knowledge on technical issues and technical people may not understand how business works as well. In our research, we want to explore more into this problem and try to propose a solution.

Our research starts with Zachman framework in which Enterprise Architecture is divided into sub architectures, among which business architecture and information system architecture come to our eyes since they correspond to the two aspects in the issue above. From Zachman and some other articles, we found that the gap between Business Architecture (BA) and Information System Architecture (ISA) exists. In order to bridge the gap, we have to study the Business Architecture and Information System (IS) Architecture first.

In order to know what the key concepts are in both kinds of architectures, we have to know the intrinsic properties in these two architectures. A better way to do this is to try to capture the properties at a higher abstract level instead of specific architectures. With this aim, we get some hints from articles that reference architecture is a direction. With the help of reference architecture, we can stand at a higher abstract level to examine both kinds of architectures and find the important concepts inside them. In both kinds of architectures, architectural pattern is found as a very important concept. Another concept is architectural perspectives. Thus, we have got an idea that since pattern and perspectives are common concepts in two types of architectures, it is possible that we could bridge the gap with the help of them. Then, analysis on the possibility of establishing linkage based on architectural perspectives is made. We found that it is possible to make some proposals for some perspectives. For the others, it is still difficult.

Then we turn to focus on studying patterns. During literature research, we found that the current business patterns are very close to complex business functions and IS patterns are too close to technical terms, more specifically, software. That also indicates the gap is really big and it is hard to link the patterns directly from the existing BA and ISA patterns.

In the latter part of this report, some changes to the current IS patterns are proposed because they are too close to software techniques and we can even call them software patterns. Since the gap is very big, we propose another kind of pattern inserting between business patterns and software patterns. This proposed kind of pattern can be called Business Information System (BIS) pattern. On one hand, it is connected with business functions; on the other hand, it is connected with software applications. We don’t give a

specific design for a BIS pattern but we give proposals what properties and components a BIS pattern should contain. Furthermore, since we propose a new kind of pattern, we also propose a new kind of architecture which is called Business Information System Architecture (BISA) in which BIS pattern is an important concept. We don’t have explored more in this direction due to time limitation and topic. However, we do propose some further study on it. Finally, the gap between BA and ISA can be somewhat smaller with the help of our proposals.

Contents

1 Introduction

Sometimes, the term “Enterprise Architecture” refers to that group of people responsible for modeling and then documenting the architecture. Other times, the term denotes the process of doing this work [1]. More commonly, when we are referring to the Enterprise Architecture, we are referring to the models, documents, and reusable items (as components, frameworks, objects, and so on) that reflect the actual architecture [49].

However, in the EACommunity (www.eacommunity.com), Enterprise Architecture is a framework or “blueprint” for how the organization achieves the current and future business objectives. It examines the key business, information, application, and technology strategies and their impact on business functions. Each of these strategies is reflected in a corresponding architecture and Enterprise Architecture is the glue that integrates each of these architectures into a cohesive framework [1].

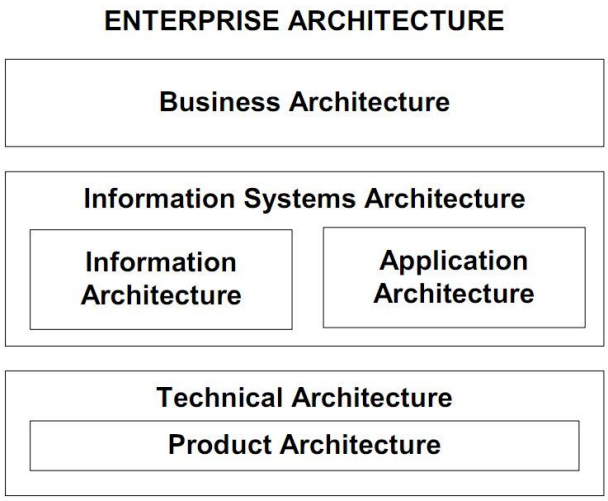

Fig. 1. Architecture Relationship [3]

Another famous resource about Enterprise Architecture is the Zachman framework. According to Zachman framework, Enterprise Architecture (EA) is a framework or “blueprint” for how the organization achieves the current and future business objectives. It examines the key business, information, application, and technology strategies and their impact on business functions [1]. Also, in the Zachman framework, the framework divides the Enterprise Architecture into 5 rows, Scope, Business Architecture, System Architecture, Technological Architecture and Detailed Representations [50]. Each row in the framework must be aligned with the cells immediately above and below it which means that the connection is necessary within Enterprise Architecture. Zachman also points out that there is a gap possibly between BA and ISA when people align business into information system. How could they do that? Who are the main responsible persons to do that? What are the key factors [2]? In [51], the author also points out that many IT

serves businesses have recognized this problem between the desire to address business issues and problems (reduce cost, create new business models, implement new business processes, etc.), and the ability to implement successful solutions using information systems. Sometimes, the gap arises because of inability of business people and technical people to communicate effectively with one another. Sometimes the technology works, but the organizational factors are not aligned with the change. Sometimes the organizational factors are in place, but the technology is not delivering as promised. Sometimes the technology works and the organizational factors are in place, but unexpectedly the needs of the business or other external factors have changed.

There will always be communication gaps between people who wake up every morning worrying about how the corporate strategy is being enacted, and people who stay up all night trying to assure that a legacy system closes within the monthly billing cycle window. The fact is, the gap may never disappear, and may actually grow [51].

Besides, in [4], Ricardo also mentions that the project of design and implementation of an Integrating Enterprise System is an extremely complex project that involves different technological, human and organizational elements. The gap between business architecture and information systems architecture makes it complex [4]. The problem is described similarly.

From these references, we could know that the gap between business architecture and information system architecture indeed exists. For this purpose, several different enterprise reference architectures (RA) have been proposed. However, this area of research is not yet totally satisfactory because these RAs may still be improved [4]. Some other studies are also conducted which indicate that this field urges to be studied as well. In [6], a GRAI integrated methodology is introduced and in [7], even companies are established by providing services to help solve the integration problems. Booz Allen Hamilton is a leading strategy and technology consulting firm, can help organizations to create, communicate, visualize and apply architectures to support decision making and achieve IT transformation. From this information, we may infer that for organizations, the gap needs a lot of efforts to be solved and a better solution is needed.

In order to bridge the gap between the Business Architecture and Information Systems Architecture, normally, we should start with studying the concepts of these two kinds of architectures. However, when we do the research, we found that there are so many specific architectures in both domain and it is hard to find something in common directly in these architectures. For example, when we examine an ISA for networked sensors in [55] and an ISA for hierarchical control of large FMSs in [56], we found that they are in two different domains. In both two architectures, we found that there is a domain model. It is good to find such a common part. However, they are in different domains. Then, we

think that it may be possible that a domain model is an important part in ISA. But if we want to make sure of it, we have to research many ISAs in different domains. Obviously, it is not a wise option to obtain important concepts of ISA and BA from thousands of specific architectures. In order to solve this problem, we turn to consider the concept of Reference Architecture. In [54], the benefits of using reference architectures are given. It captures the main ideas and components across domain and it provides a higher level abstraction for architecture itself. That means we don't have to reinvent the thousands of specific architectures one by one. Generally, a reference architecture in the field of software architecture or enterprise architecture provides a template solution for an architecture in a particular domain. It provides a general design for specific architectures with capturing its common properties. By studying RAs, we are possible to know what key concepts are contained in the two kinds of architectures respectively [5]. As mentioned above, many different enterprise reference architecture have been proposed which means that RA is an interesting direction to work with for a solution.

In the next chapters, we will first discuss the definitions of RAs in both domains. Then we try to find the commonly important concepts. After comparing similarities and differences, we try to propose a framework to shorten this gap by making use of the similar key concepts of the two types of RA's.

In our research, basically, we follow the Kitchenham systematic review guideline [8] and Budgen systematic literature reviews in software engineering [9].

2 Elaboration of research questions

Considering the given background above, we elaborate our research questions as follows. Given the goal to bridge the gap between business architecture and information system architecture, our primary question should be as follows:

- What are key concepts in business architecture and information system architecture, are there any interrelations? How the gap could be bridged with the help of these interrelations?

To find the answer to this question, we need to solve some questions about RAs first as we discuss already in the introduction:

- How is Information System Reference Architecture defined?
- How is Business Reference Architecture defined?
- What are the differences, what is the gap?
- What is in common appearing in the definitions, which (similar) concepts are being addressed in these definitions and could they be of use to bridge the gap?
- How to identify and to explore ways to bridge the gap?

After getting the answers of above questions, it is possible that from the common important parts in both RAs, we can find a hint to bridge the gap mentioned above. Finally, a framework to bridge the gap between BA and ISA by making use of key concepts is developed.

3 Methodology

Basically, our research is a literature study. According to [8], the following aspects have to be considered. In our study, we plan to take several turns of studies. New research questions would be asked during study process.

Data Sources

Initial searches for primary studies can be undertaken initially using electronic databases (mainly from Google scholar) but this is not sufficient [8]. Other sources of evidence must also be searched (sometimes manually) including:

- Reference lists from relevant primary studies and review articles
- Journals (including company journals such as the IBM Journal of Research and Development), grey literature (i.e. technical reports, work in progress) and conference proceedings
- Research registers
- The Internet

Study Selection

Once the potentially relevant primary studies have been obtained, they need to be assessed for their actual relevance.

Included studies include all the articles related to RA concepts such as architecture patterns, architecture model and other perspectives. However, if a literature is simply talking about a RA in a specific domain without referring to RA concepts effectively, we exclude it. Literatures talking about different architecture types are included and reviewed as well because our topic is about two types of architectures.

Quality Assessment

We have a loosely assess to the literatures we obtained. After reading the abstract, we can have an opinion on which aspect it is addressing. If it is all about solving our primary and sub questions, we pay much more attention and leave more time on it. However, as to the literatures that introduce a concept or give an example, we focus on the concept or the example and have a rough glance at other part.

Data extraction

After reading each literature, the data extraction forms must be filled in. In our research, it is a doc document collecting all the information needed to address the research questions. In addition, to include all the questions needed to answer the review question and quality evaluation criteria, the documents should provide standard information including Date of Data extraction, Title, authors, journal, publication details and space for additional notes.

Data Analysis and discussions

The next step is to analyze the data and based on the research questions, try to give answers.

Secondary search

After primary studies about definitions of RA is done following the above four steps, some answers to sub questions might be found. Based on that, a second turn searching of specific terms (important concepts in both RAs) is conducted. The Data sources, Quality Assessment and Data extraction are the same with primary search. As to the study selection procedure, we only select those literatures that highly focus on the terms we are interested in without talking other elements included. Also, a data analysis and discussion is done.

More studies

If necessary, more studies and researches are done in a similar way with the first two turns of study.

Elaborating of new ideas

Having obtained enough information, a new idea or model should be proposed to solve the problems about the interrelating of information system RA and business RA. If there is no ideal solution to this problem, we have to admit and say that in the report.

Making conclusions

In this step, we conclude our study and also talk about the deficiencies of our study. Based on that, we also suggest the future study directions.

4 RA definitions

According to the schedule, first we will search for the current articles or books about definitions of RA for information systems and RA for business. The references are from [10] to [22] we found from resources. Then we will try to find out important parts of them two.

In this chapter, we are dedicated to solve these research questions:

- How is Information System Reference Architecture defined?
- How is Business Reference Architecture defined?
- What are the differences, what is the gap?
- What is in common appearing in the definitions, which (similar) concepts are being addressed in these definitions and could they be of use to bridge the gap?

4.1 Definition of Information System RA

Existing literatures have loosely defined what an Information System RA is. Some examples are as follows:

> *"A Reference Architecture is a general design (abstract blueprint) of a structure for a specific class of information systems* [10]*."*

In this definition, the author analogies RA as an abstract blueprint. He also indicates that RA is a general design of structure for a specific class which in our opinion, should also have similar mapping in the definition of business RA. Like information systems, business architectures vary according to their business functions.

> *"A Reference Architecture is generic software architecture for a class of software systems that is used as a foundation for the design of concrete architectures of systems from this class* [11]*."*

In this definition, it defines the function of an information system reference architecture. It should be a basis that can be used for the future design.

> *"A Reference Architecture: …is, in essence, a predefined architectural pattern, or set of patterns, possible partially or completely instantiated, designed, and proven for use in particular business and technical contexts, together with supporting artifacts to enable their use* [12]*."*

This definition gives more details and terms which allow us to think deeper. It talks about the concept of pattern and pattern is predefined. It is also possibly containing not only a pattern but also set of patterns. It gives a more general idea about the possible areas that it can be used both in business and technical domain respectively. This information is interesting for us because we are finding the similarities between business RA and information system RA. In this definition, pattern might be a key concept.

- *"A Reference Architecture is an elaboration of mission, vision and strategy* [12][13]*."*

This definition is also interesting that it stands at a higher level perspective. In our opinion, this conception doesn't distinguish clearly between business RA and information system RA. Both of them could be taken as an elaboration of mission, vision and strategy. We will consider this character when we map the two kinds of RAs because this definition gives us a hint that although specific structures may differ in both architectures, they could still have similarities in accordance with their mission and strategies.

- *"A Reference Architecture models the abstract architectural elements in the domain independent of the technologies, protocols, and products that are used to implement the domain* [14]*."*

Here, an information system RA models the abstract architectural elements, which means that RA might contain more abstract elements. From here, we have a question: if a RA contains abstract elements, how abstract it should be? This impacts our goal in the basic sense that we should link the elements of two architecture on the same abstraction level. Then another question appears: except for abstraction level, what other architectural perspectives we should consider?

- *The Reference Architecture takes the reference model in information system domain as its starting point in particular in relation to the vocabulary of important terms and concepts.*

Here, the concept of reference model is mentioned. RA starts from it base on a specific domain. Actually, this concept is mentioned many times in several articles although they don't always appear in the definition of RA.

- *"A Reference Architecture provides a template, often based on the generalization of a set of solutions. These solutions may have been generalized and structured for the depiction of one or more architecture structures based on the harvesting of a set of patterns that have been observed in a number of successful implementations* [15]*."*

Again, pattern's conception appears again which makes us believe that pattern is an important term in information system RA.

- *"A Reference Architecture is the generalized architecture of several end systems that share one or more common domains* [16]*."*

Here, words like "generalized" and "common" indicate its repeated property.

- *"A Reference Architecture is a way of documenting good architectural design practices to address a commonly occurring problem* [17]*."*

This explanation also contains the word "commonly" again. This definition, also gives a more general concept of RA in documenting good architectural design. This common property may also help our study.

From almost all of these conceptions, we have an understanding that RA is an architecture with the help of which concrete architectures could be designed more easily. In some articles, we noticed the concept of Reference Model. We should notice that there are differences between the conception of Reference Model and RA [14]. The primary contribution of reference model is that it identifies the key characteristics of a specific system, and it defines many of the important concepts needed to understand what the system is and what makes it important. The RA takes the reference model in information system domain as its starting point in particular in relation to the vocabulary of important terms and concepts. Consider the time of mentioning, we think reference model should be an important concept. Since the concept of pattern is mentioned many times as well in literature ([10], [11], [12],[14]) as an import element, we have enough reason to regard it as an important concept. Besides, according to our analysis, we come up with two other questions about architectural review perspectives. To understand complex information system architectures, it is necessary to be able to look at these architectures in a well-structured way. It is not wise to link the two levels of RAs at different abstraction levels. A study of architectural perspectives is necessary as a basis to link the two kinds of architectures.

In summary, a definition of Information System RA summarized from above literatures can be as follows:

- A Reference Architecture (RA) for information systems is a generic architecture for a class of systems that is used as a foundation for the design of concrete architectures of systems from this class. It combines the concept of patterns on the basis of a IS reference model and recurring of design becomes possible.

Although in this definition, architectural perspective is not mentioned, architectural perspective helps us in making appropriate choices with respect to looking at architectures and describing them [10] and as a basis of linking reference. We will discover more about it later. Considering all, we can know that for information system reference architecture, patterns and reference model are important key concepts. We should also attend that both patterns and reference model are based on a perspective framework which is also a very important conception. We say this because we consider linking RAs by linking key concepts on the same perspective or we can also linking by specific perspective(s). For example, if stakeholder is a perspective, we can compare the different stakeholders in both kinds of architectures and try to establish connections. We will give more details later in this report.

In the next chapter, according to our schedule, we will study the definition of business reference architecture and its related key concepts.

4.2 Definition of Business RA

Although the term "Business Architecture" is used in numerous publications, the concept is not defined unambiguously [18].

- Wolfenden and Welch use *"business architecture" as the connecting link between strategies on the one hand, and business processes, roles, behavior and information on the other* [19].

In this definition, strategy is a core concept.

- A business architecture is also defined as *a part of an enterprise architecture related to corporate business, and the documents and diagrams that describe that architectural structure of business* [20].

Here, an architectural structure is mentioned. From [10], architectural structure includes concepts of patterns, styles and dimensions.

- In [21], *a business RA starts from business concepts combining with enterprise requirements. Through matching basic business concept patterns and custom development, enterprise solutions are achieved finally.*

Here, business concept patterns are mentioned again. Enterprise solutions are achieved through matching basic business concept patterns and custom development [21].

- *Business architecture bridges between the enterprise business model and enterprise strategy on one side and the business functionality of the enterprise on another side* [20].

Here, business model and enterprise strategies appear again.

- *Business Architecture that describes how the enterprise needs to operate to achieve the business goals, and respond to the strategic drivers set out in the Architecture Vision, in a way that addresses the Request for Architecture Work and stakeholder concerns* [22].

Here, the business architecture is described to respond to the strategic drivers in an architecture vision and to satisfy stakeholders' concerns. Stakeholder's concept is interesting both in information system RA and business RA and it should belong to one of the perspectives that we will discuss later.

In summary, a definition of Business RA summarized from above literatures can be as follows:

- A business Reference Architecture is a generic architectural for a specific business goal. Starting with business model, combining with Business patterns, under regulation of Architectural Perspectives, a business RA is serving and reflecting its business strategies and as a basis to discovering more subsequent architectures including information system architecture and technological architecture.

As a summary, we know that for a specific BA, patterns, architectural perspectives and business strategies are three important concepts that appear many times in references. In chapter 4.3, we will discuss more about the common important parts in both definitions of business RA and information system RA.

4.3 Important concepts in both RAs and differences

In order to inter-relate the two types of reference architectures, we can find some characteristics they have in common based on the definitions.

From the finding results, we are safely to say that architectural patterns are important concepts in both types of RAs. There are business patterns in business RA. Accordingly, there should exist corresponding information system patterns. Otherwise, the enterprise reference architecture might fail to connect its business architectural level and information system architectural level. As a result, our study will contain the concept of patterns. Another thing is that in information system RA, we found that abstraction is an important word. In business RA, stakeholder is an important concept. On the surface, they don't have any relationships. However, from [10] and some other articles, they both belong to the architectural perspectives. Thus, we think architectural perspective is an important concept as well. As to the possibility of bridging the tap through architectural perspectives, firstly we need to do more researches on architectural perspectives in the chapter 5. After that, we will give further analysis.

As to the differences, in business RA, business strategy is stressed. However, in IS RA, no concept is related to business strategy. This means that the business strategy can't be aligned directly to the concept in IS architectures. Another difference is that in IS architecture, it is IS reference model oriented. However, in BA, it is business model oriented. To some extent, the intrinsic different between these two kinds of models leads to the gap mentioned above. The differences between IS architecture and BA all make the gap larger. If we want to bridge the gap, we should consider start with the common concepts we found, patterns and perspectives respectively. A good thing is that in the above definitions, we also know that both the IS RA and business model involve the concepts of patterns and perspectives. Pattern can be seen as a recurring structure in a model [17] and perspectives are used to view a model. Thus, it is possible that through studying patterns and perspectives, the differences between IS reference model and business model could be smaller.

In summary, in both architecture level and model level, patterns and perspectives are two common important concepts. We will do more researches on these two concepts to see whether they are able to be used to bridge the gap.

To here, some questions have already been answered:

- How is Information System Reference Architecture defined?
- How is Business Reference Architecture defined?
- What are the differences, what is the gap?
- What is in common appearing in the definitions, which (similar) concepts are being addressed in these definitions?

New questions appear as follows:

- What are the architectural perspectives? What are important ones to us?
- What are the patterns in information system architecture? How do they function? Why do we use them?
- What are the patterns in business architecture? How do they function? Why do we use them?
- Can perspectives and patterns be used to bridge the gap?

Starting from Zachman's framework and the study of RAs, we know that the gap exists between BA and ISA and the reason behind it. With the help of business RA and information system RA, we also know the important concepts in specific BA and ISA since that RA is a general design for specific architectures. Having captured these concepts, we turn to focus on architectures rather than RA more in order to find a way to bridge the gap. All unsolved questions including our primary question will be studied in the next several chapters.

5. Perspectives and patterns

From our preliminary research, we have the conceptions of RAs for business architecture and information system architecture. From these two conceptions, we know that patterns are important concepts for the design of the two architectures. In this chapter, we will discover more information about architectural perspectives and patterns. In this chapter, we focus on these study questions:

- What are the architectural perspectives? What are important ones to us?
- What are the patterns in business architecture? How do they function? Why do we use them?
- What are the patterns in information system architecture? How do they function? Why do we use them?

5.1 Architectural perspectives

To understand complex information system architectures, it is necessary to be able to look at these architectures in a well-structured framework. Without a framework, it is impossible to analyze or design complex, multi-model architectures. [10] Different authors use different terms for architectural perspectives, like architectural viewpoints or architectural dimensions [10][24][25] for different purposes. In this chapter, we examine some perspectives and to see whether they can help us build the final framework.

Aspect:

A classification of system properties or aspects as they are also called describes a number of aspects from which we can view an architecture, providing a certain "cross-section" of an entire architecture description [10][24]. One single architecture can be looked at from different aspects or an architecture focuses on only one aspect and others are not emphasized. According to different focuses, proper models should be chosen including the characteristics that are most concerned. In order to give a better opinion about the conception of aspect, we can refer to building architectures. In the architecture of a house, for example, one can distinguish easily between external architecture, the interior architecture and infrastructure architecture. Different building architectures have their special models. For example, the infrastructure architecture describes how the plumbing and electricity circuitry is organized. However, the external architecture focuses on the externally visible structure of the same house.

Abstraction:

In the abstraction perspective, we decide how abstract or concrete an architecture description needs to be. It indicates the gap between the RA and a solution architecture

based on it. [25] In most cases, especially those for top management, architecture doesn't have to be highly concrete because they are used to provide an overall perspective of a system or as a basis to design concrete architecture which is actually much more concrete. However, it shouldn't be completely abstract as well since it means no concrete information is given and it turns out to be worthless. This perspective especially deserves to be considered for an architecture designer because he or she should consider the proper abstraction level seriously given a customer context and requirements. From [26], some architectural examples about different abstraction levels are given.

Figure 3 shows a simple RA that is very abstract because it only shows a structure but no information of the functionality of the components. Although it specifies that there are four components and how they are interconnected, it is not so useful in practice. Consequently, it would be on level 0 which is the lowest level according to Truijens' standard. [26]

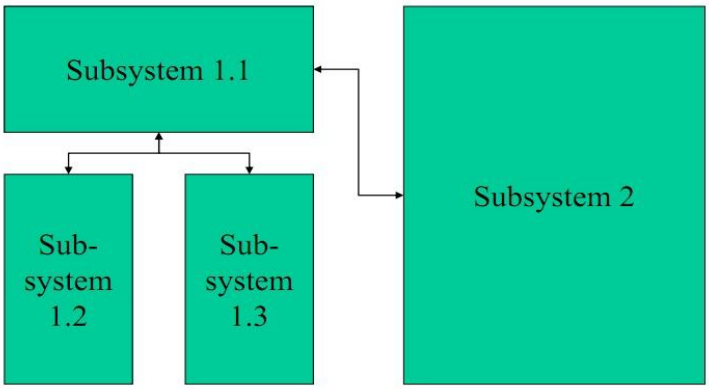

Fig. 2. Example architecture with abstraction level 0

Figure 3 shows an architecture for the same system but with more specifications. It is regarded at abstraction level 1. At this level, each component has an indication of the software system class used, hence making the RA more concrete than that at level 0.

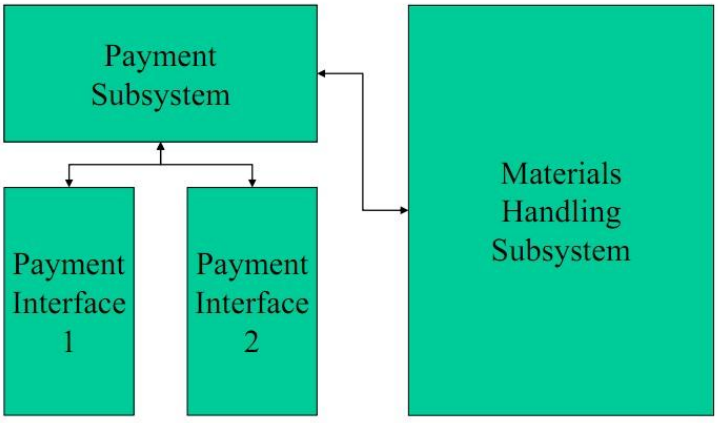

Fig. 3. Example architecture with abstraction level 1

Figure 4 shows an architecture for the same system again but one step further in abstraction level. In this architecture, the corresponding sub-systems are replaced with specific software system type based on the architecture in figure 3. [10] Obviously, more and more information is included as the abstraction level goes up. From these primary examples, we can understand that different abstraction levels contain different amount of information which requires the architecture designers to choose abstraction levels appropriately. As a result, a good design can not only satisfy customers' needs, but also can avoid more redundancy.

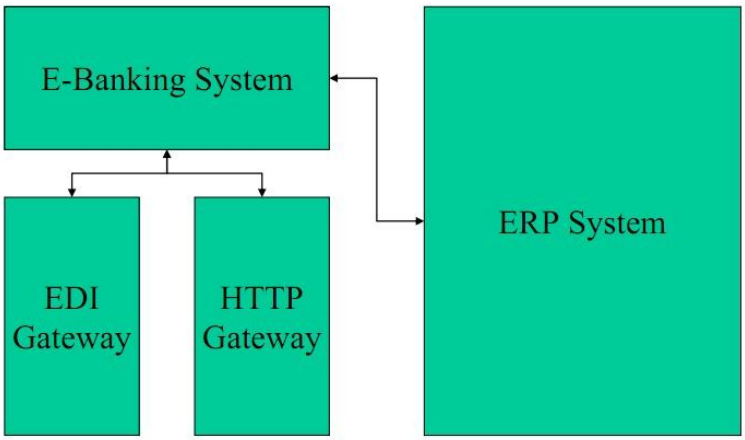

Fig. 4. Example architecture with abstraction level 2

Aggregation:

The aggregation perspective allows higher-level concepts to be broken down into subcomponents. [27] It determines how detailed an architecture model is with respect to the number of components identified. [10] Aggregation criterion differs from abstraction in the sense that moving in the aggregation level simply means changing the number of items in an architecture while the level of detail of each item remains the same. A higher aggregation level means bigger number of components. We can see the next example.

For the same information system, the RA in figure 5 is with a lower level of aggregation criterion. It only contains one component and doesn't provide much information. However, the RA in figure 6 divides the whole system into two sub-systems which, as a result, has a higher aggregation level. It gives the idea that the big system can be separated into two small and parallel systems and apparently, more information is involved. Functionally, both aggregation criterion and abstraction criterion serve for how much information a RA could give in terms of number of components contained and level of detail of components.

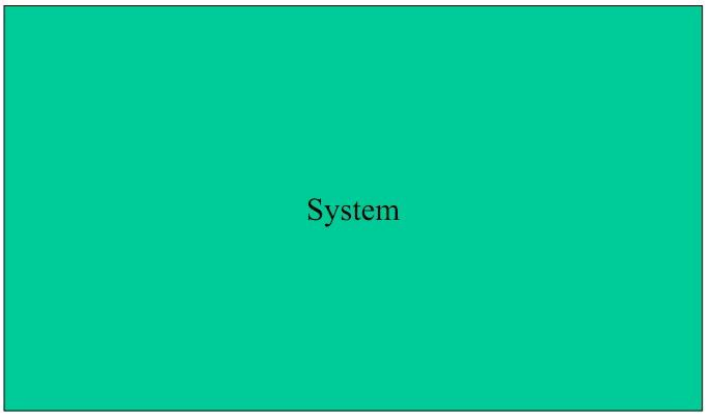

Fig. 5. Architecture with a lower aggregation level

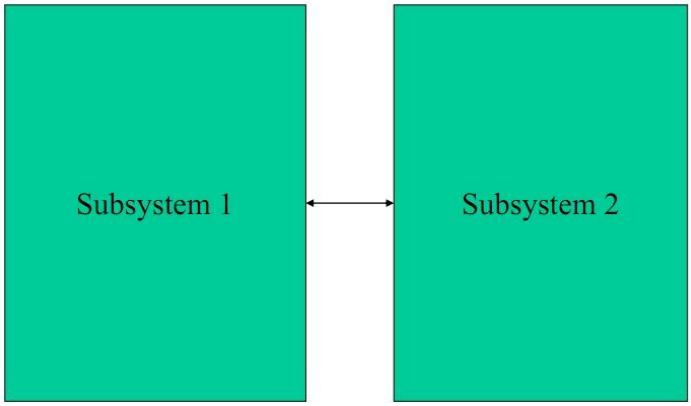

Fig. 6. Architecture with a higher aggregation level relative to figure 6

Business or Technology oriented:

Architectures are widely used nowadays not only in business information system domain, but also many other domains where architectural thinking is absorbed. Generally, there are two main orientations of an architecture focus. An architecture could be highly business oriented, for example, for a business organization, or highly technology oriented, for example, for adaptive hypermedia applications [28]. An architecture could also be somewhere between the two extremes, for example, a RA for ERP system [30] or e-commerce system. [29] The business orientation requires architecture designers to answer questions like whether required business functions are supported, whether the designed architecture keeps the organization competent compared with its competitors et cetera. On the other hand, the technology orientation requires architecture designers to think fully about the properties of a complex technology system, such as compatibility, consistency, efficiency and updateability et cetera and it focuses more on technological realization, possibly including software, hardware, language and protocols.

Different abstraction and aggregation level should be considered according to different orientations. This is partially because of the fact that business people may not have

sufficient technical background or just don't care technical details in the system and vice versa. [29]

Coverage:

Coverage indicates the area where a RA can be useful. Some RAs focus only on presentation, integration or security aspects of solutions, whereas others cover an end-to-end enterprise solution. What would lie at the extremes of this viewpoint? At the narrow one, an Architecture Pattern which is consistent with the view that a RA is an aggregation of Architecture Patterns. At the other, we should expect end-to-end architecture that covers both business and IT solutions. [31]

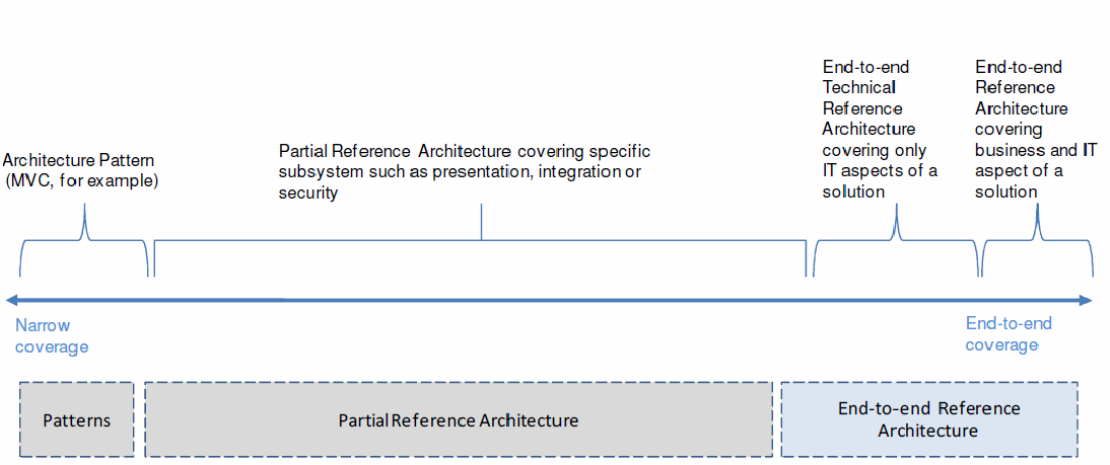

Fig. 7. RA coverage ranges from very narrow (Only Architecture Pattern) to full end-to-end that includes both business and IT aspects of a solution

By examining a RA from coverage point of view, we could have a better understanding about its function, whether it is for presenting something or focusing on integration aspect or both. For example, in [31], a RA for monitoring and control system focuses the system integration other than security and presentation. It adopts several strategies so that components and connectors are used and connected closely. Although this RA employs UML as modeling language which is quite expressive, presentation is not the primary consideration. As a result, it is not an end-to-end RA but a partial RA that focusing on system integration. In [30], a service-oriented RA is introduced and it includes both business and IT aspect of the SOA solution even if it is a bit biased to IT aspect. Therefore, it is more or less close to end-to-end coverage level.

The coverage viewpoint differs from aggregation viewpoint. A RA could be at a high coverage level which means it covers issues for both business and IT aspects but with a relatively low aggregation level which means the number of components is still small. (Fig. 9) From the RA given in the figure below, we can know that its abstraction level is not so high as well.

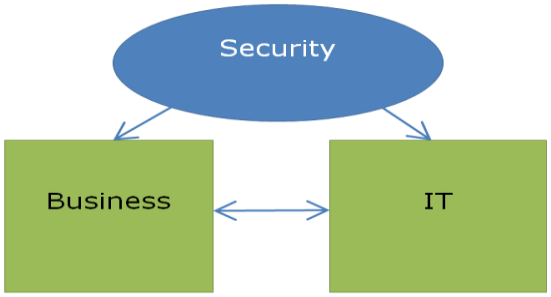

Fig. 8. A RA at a high coverage level with a low aggregation level

Stakeholder:

A successful architecture has to reflect the concerns and interests of the stakeholders. [32] In [33], architecture is described as "a vehicle for communication and negotiation among stakeholders". Taking that into account, the architecture must also reflect the different viewpoints of all the interested parts, so that it can be communicated efficiently.

Also in [34], a stakeholder is defined as a viewer that perceives and conceives the universe, using his/her senses, in order to produce conceptions resulting from the interpretation of what is observed. Specifically, a stakeholder could be any of these sorts listed below according to literatures [33] and [34]. Other stakeholders possibly related to RA review are planner, builder and programmer et cetera.

Stakeholder	Description
Domain Experts	People who have a deep insight about the topic related to the RA
End User	People who uses the application or function described in the RA
Application Developer	People who develop the application according to the RA
System Integrator	People who is responsible for system integration
Manager	People who views the RA from management point of view

Table 1. Crucial stakeholders related to RA review

To fully consider every stakeholder possibly related to the architecture design is important in the sense that they are the real architecture readers. Basically, the architecture designer should understand what the stakeholders need and try to satisfy their interests as much as possible. In addition, the architecture designer should also consider their background so that the result architecture is applicable rather than unreadable. Besides, special requirements could exist sometimes, which requires the architecture designer to treat the situation flexibly and differently.

5.2 Business Architecture Patterns

(1) Definition

A business pattern is a common design that can be executed repeatedly by various business models in different environments [35]. A pattern is discernible coherent and predictable series of related acts, interactions or events based on specified relationships that can be imitated, and represents an established mode of behavior [35]. A pattern describes a dynamic behavior that occurs repetitively and which can be used to create value [35]. A business pattern is characterized by being instructive, structural, reusable, proven and lastly as making business sense [35]. Besides, patterns do not substitute for design experience and skill [36]. After all, we don't necessarily want to see a series of identical, mass-produced porches along a street – we want to see a series of individual porches, but with some deeply shared design principles. [36]

According to [35], a particular industry might show preference for a specific business pattern because it offers some unique benefits to an industry. From here, we know that business patterns vary a lot with different industry domains. In our research, we aim to build some kind of relationship between business patterns and information system patterns. It is unrealistic that we analyze all business patterns in all industries. However, we focus on the very basic business patterns that could exist in all industries.

A classification schema of business patterns will help managers to understand why business's change and what business benefits the change need to bring. [35] In the next chapters, we study basic business patterns in a classification schema.

(2) Business pattern classification

Generally, business patterns can be classified into two categories, business structural patterns and business behavioral patterns. [36]

A structural design pattern serves as a blueprint for how different classes and objects are combined to form larger structures. Each structural pattern has a different purpose. [38] More specifically, there are two kinds of structural patterns, policy level structural patterns and operational structural patterns. [37] Some example structural patterns are Group pattern, value chain pattern, contract pattern etc. [37]

Behavioral design patterns are design patterns that identify common communication patterns between objects and realize these patterns. By doing so, these patterns increase flexibility in carrying out this communication. [39] Some examples are Account pattern, Posting pattern, etc.

In the next picture (Fig. 10), we give a picture to represent classification of business patterns. This picture is concluded from articles [37] and [38] and it gives a direct classification. There are surely more patterns waiting to be discovered and also patterns

for specific industries. However, what we focus on here only includes the basic existing patterns that can almost serve every industry [37] [38].

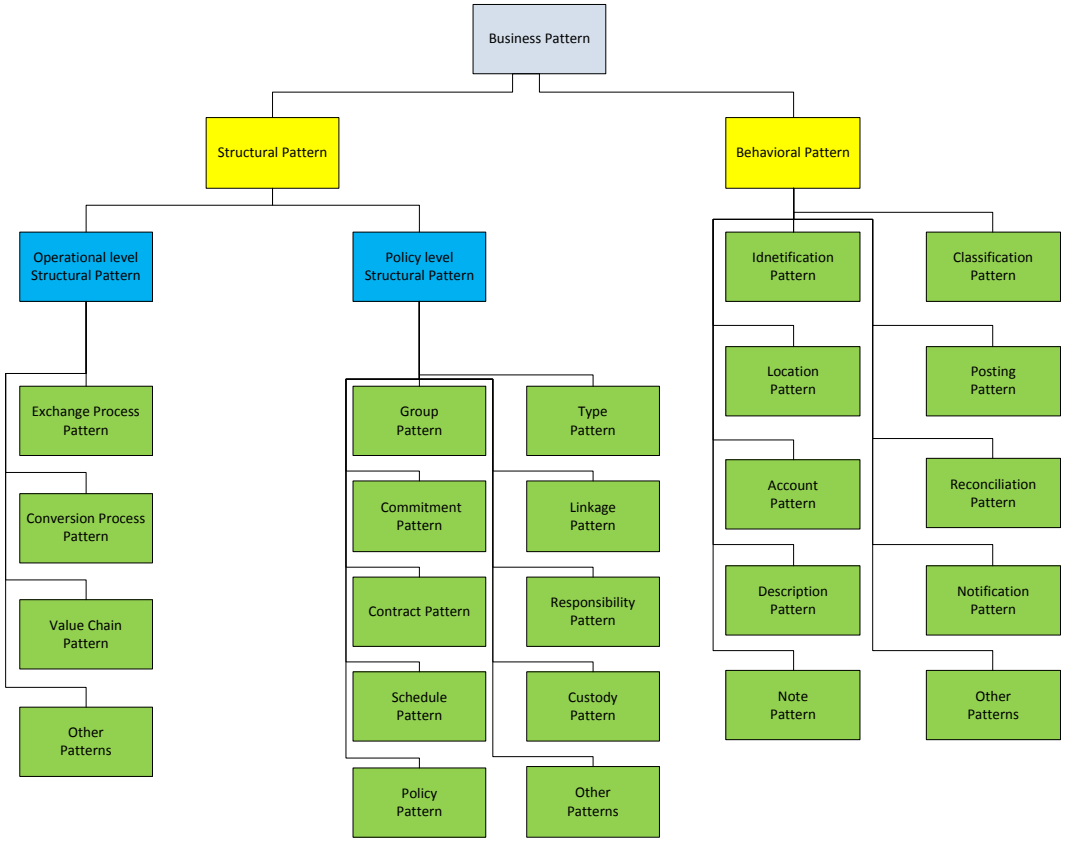

Fig. 9. Business Pattern Classification (summarized from [37] [38])

In this business pattern classification, generally, it contains two kinds of big categories, structural pattern and behavioral pattern. Under structural pattern category, there are two sub categories as well, operational level structural pattern and policy level structural pattern. The green blocks indicate the very basic patterns under each category. Since that it has lots of basic patterns, it is impossible to list all of them. Besides, it is also meaningless and time consuming to introduce each one with lots of details due to our aim that we don't focus on a specific pattern but an overall bridging approach. In the next part, we select several of them and give a general introduction about them.

Group Pattern:

Business rules seldom refer to a specific instance, such as an actual customer or an actual item [37]. For example, a shop gives a 20% discount to the customers living in a specific geographic area. In principle, the manager can create an individual discount rule for every customer from this area. However, if the manager has 100 customers from this area, then the business application would contain 100 rules, and they would all be the same. This is possible, but impractical [37]. A better solution is to have one rule, and apply it to the entire set of these customers. This is so-called group patterns.

The purpose of the business rules is to give general guidelines that are applicable to groups of economic resources, events, and agents, rather than to actual resources, events, and agents [37].

Type Pattern:

Product catalogues often contain a description of the resources that a customer can buy, rather than actual resources. When customers place an order, they specify the parameters of the product; when a vendor successfully fulfills the order, he delivers an actual product that conforms to the parameters of the customer's order. A similar example can be told for production. A recipe or blueprint contains the parameters, description, or definition of a production is successfully completed, products that match the blueprint are produced.

If you design a business application, you often need to create a model that contains catalogue-like descriptions of resources, events, and agents [37].

Basically, types are homogeneous groups; all their members conform to certain definitions, descriptions or blueprints.

Commitment Pattern:

Most economic events do not occur unexpectedly. Economic events are usually scheduled or agreed upon beforehand by economic agents. For example, a sales order line is a promise to sell goods to a customer; the total price is a customers' promise to pay for the goods, and the seller's promise to accept the payment.

Commitment pattern solves the problem that how we model promises of future economic events [37].

Contract Pattern:

Commitments represent the optimistic path of an exchange. For example, a sales order contains commitments to deliver good and commitments to pay. However, sometimes goods are not delivered as expected and payments arrive late. Partners usually also agree upon what should happen if the initial commitments are unfulfilled. However, sometimes it is not so optimistic. A contract defines everything based on rules no matter whether the commitments are fulfilled or not. Penalties and rewards might be included as well.

Generally, this pattern defines what would happen if the commitments are not satisfied [37].

Schedule Pattern:

Production processes usually do not occur spontaneously; a rational company schedules the production and usage of its resources that should take place in the future. However, production sometimes does not occur as planned because of unexpected circumstances. A rational company would like to mitigate risks and determine additional factors that should occur if the originally planned operation does not occur as expected. Making a plan is a way to minimize the risks of missing some resources in the middle of a production. The

purpose of the plan is to make sure that for all processes the needed resources are identified, as well as when they will be needed [37].

In short, this kind of pattern deals with how we specify conversion processes that should occur in the future.

Identification Pattern:

People refer to real or imaginary things by their names. We name things to identify them, so we can refer to them by their names and not just point to them and say "this!". By naming, we give things identities, but in real life they are not often unique. Many things have more than one name, and sometimes a single name can refer to different things, which is fine as long as everyone who uses that name knows what thing it refers to. In business, people use serial numbers, production numbers, civil registration numbers, and names.

This kind of patter solves the problem that how we specify the identity of things.

Location pattern:

Most economic events take place in time and space. For some economic events, the location is an essential attribute characterizing them. Shipment, for example, is an economic event in which an economic resource is moved from one location to another. Users of business application are interested not only in departure and destination, but often also in the actual location of the economic resource during the economic event.

This kind of patterns solves the problem how we specify the place where the economic events occur.

Posting Pattern:

Registering the history of the realized or intended exchanges of economic resources is an important part of the functionality of most business software solutions. For example, when economic agents agree upon a commitment, the commitment should be registered, and all modifications to the registration should be traceable. Tax authorities often set similar requirements for exchange economic events and for materialized claims; any changes made to the data that influence the economic results of the company should be traceable.

The history of business relationships is typically related to realized or intended exchanges of economic resources, contracts, agreements, and claims, such as the purchase and sale of products and services, invoices, and corresponding payments.

How do we keep track of the history of economic events, commitments, contracts, and other entities that represent interactions between economic agents? This kind pattern aims at solving it.

Account pattern:

The posting pattern describes how to record the history of economic events, commitments, contracts, and claims. However, merely keeping track of these entities is usually not the main interest of an enterprise's decision makers. Users of business applications are also interested in aggregated information about the state of the enterprise. This kind pattern stresses the questions that how we keep track of the total amount of something being increased or decreased, and eventually compute what constitutes the total amount.

Notification Pattern:

Various users of business applications should often be notified when certain events occur, or when certain conditions become true. For example, both customer and banker may be interested in being informed when the customer account has been overdrawn. Business applications can be configured to create and send notifications automatically.

The notification pattern is designed for the question: how we notify users of business applications about changes in business events or status?

5.3 Information System Pattern

(1) Definition

In [10], an architecture pattern is a generally recognized recurring (sub) structure that is used to describe part of the overall structure of an architecture. This definition points out the common property for a pattern not only for information system architecture but also for business architecture. Specifically, literature [10] tells that patterns are most commonly used for the software (application logic) aspect of information system architectures. In other words: they are used to design the structure of the software of the information system under design.

In [40], design patterns provide a common vocabulary for designers to communicate, document, and explore design alternatives. They reduce system complexity by naming and defining abstractions that are above classes and instances. A good set of design patterns effectively raises the level at which one programs.

(2) Information system pattern classification

In [10], four very basic patterns are introduced: direct invocation, file transfer, shared database and shared bus.

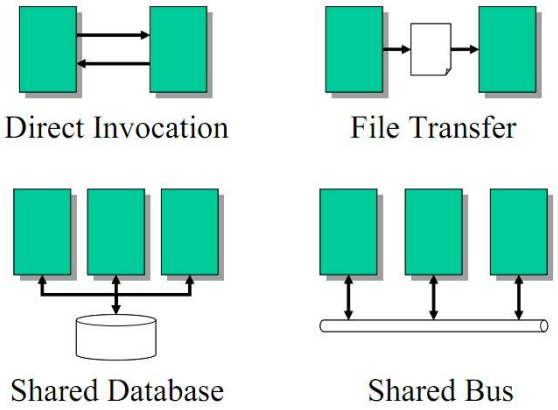

Fig. 10. Basic patterns

Also in [10], detailed descriptions are given as follow table.

Name	**Description**	**Application**	**Requirements**
Direct Invocation	Remote procedure invocation between modules	One-to-one module coupling	Modules must be active simultaneously
File transfer	File transfer between modules	Offline 1-to-n module coupling	Existence of predefined file structure
Shared Database	Data transfer via shared database	Flexible asynchronous n-to-m module coupling	Existence of predefined shared database, availability of transaction management
Shared Bus	Data transfer via bus	Direct flexible m-to-n module coupling	Existence of standardized bus, modules must be active simultaneously

Table 2. Very basic pattern catalogue

Except for this catalogue, other catalogues exist. A classical classification of software patterns is based on [41] as follows (Figure 11). It is helpful to know more details about each pattern appearing in this catalogue as well.

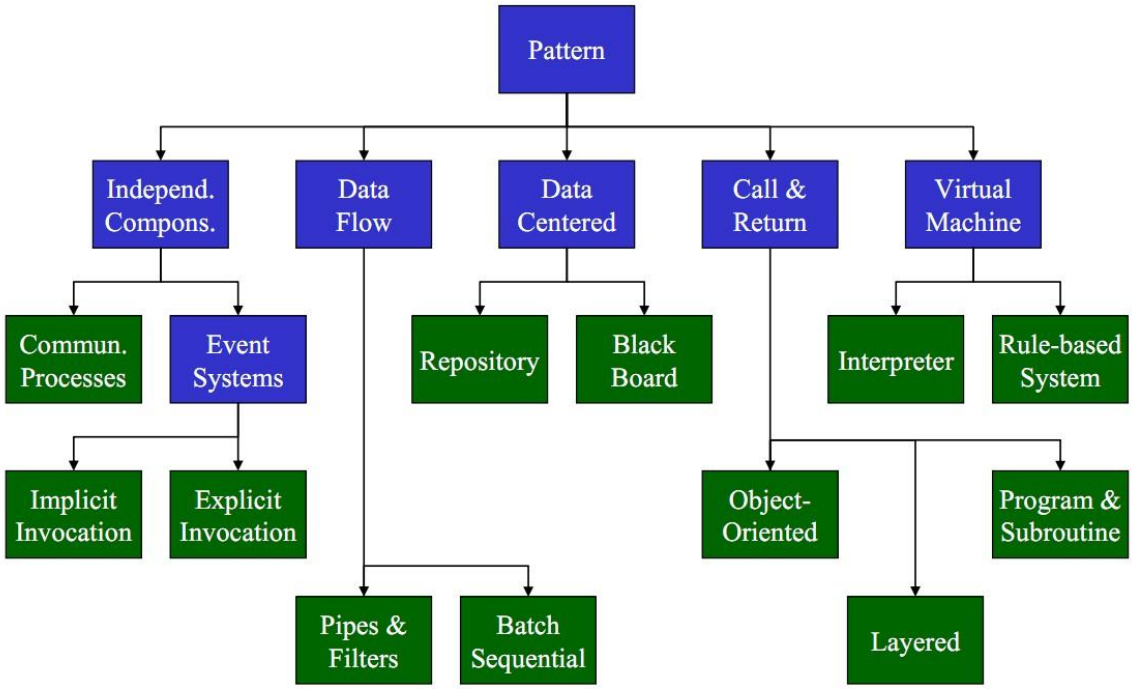

Fig. 11. A bigger pattern catalogue [41]

Implicit Invocation Pattern:

Implicit invocation is a term used by some authors for a style of software architecture in which a system is structured around event handling, using a form of callback [42].

The idea behind implicit invocation is that instead of invoking a procedure directly, a component can announce (or broadcast) one or more events. Other components in the system can register an interest in an event by associating a procedure with the event. When the event is announced the system itself invokes all of the procedures that have been registered for the event. Thus an event announcement implicitly causes the invocation of procedures in other modules [43] [44] [46].

An important benefit of implicit invocation is that it provides strong support for reuse [43]. A new component can be added into a system with simply registering it for the events of the system. A second benefit is that implicit invocation eases system evolution [45]. Components may be replaced by other component without affecting the interfaces of other components in the system.

Explicit Invocation Pattern:

Consider a component, the client, which needs to invoke a service defined in another component, the supplier. Coupling the client with the supplier in various ways is not only harmless but also desirable. For example, the client has to know the exact network location of the component which offers the service for the sake of improving performance; or the client has to initiate the invocation itself; or the client must block and

wait for its result before proceeding with its business. How could these two components interact with each other?

An explicit invocation allows a client to invoke services on a supplier, by coupling them in various respects. The decisions that concern the coupling are known at design-time. The client provides these design decisions together with the service name and parameters to the Explicit Invocation mechanism, when initiating the invocation. The explicit invocation mechanism performs the invocations and delivers the result to the client when it is computed. The explicit invocation may be part of the client and the server or may exist as an independent component as well [46].

Batch Sequential Pattern:

Consider a complex task that can be divided into several smaller tasks, which contain a series of independent computations. This can't be realized by one monolithic component because this component would be overly complex, and it would hinder modifiability and reusability.

In the batch sequential pattern, the whole task is divided into small processing steps, which are realized as separate, independent components. Each step runs to completion and then calls the next sequential step until the whole task is fulfilled. During each step a batch of data is processed and sent as a whole to the next step [46].

Batch sequential is a simple sequential data processing architectural pattern. It is useful for simple data flows, but has severe overhead for starting the batch processes and transmitting data between them. Instead, the following Pipes and Filter pattern is more useable for stream-oriented data-flow processing, where the filters incrementally transform their input streams into output streams.

Pipes and Filter Pattern:

In a pipes and filters architecture a complex task is divided into several sequential sub-tasks. Each of these sub-tasks is done by a separate, independent component, a filter, which handles with only this task [46]. A component reads streams of data on its inputs and produces outputs, delivering a complete instance of the result in a standard order [43]. Pipe and filter systems have a number of nice properties. First, they allow the designer to understand the overall input/output behavior of the individual filters. Second, they support reuse: any two filters can be connected together, provided they agree on the data that is being transmitted between them. Moreover, systems can be easily maintained and enhanced: new filters can be added to existing systems and old filters can be replaced by improved ones. Fourth, they permit certain kinds of specialized analysis, such as throughput and deadlock analysis. Finally, they also support concurrent execution. Each filter can be executed as a separate task and potentially executed in parallel with others [43]. The next figure gives an example of a system using pipes and filters.

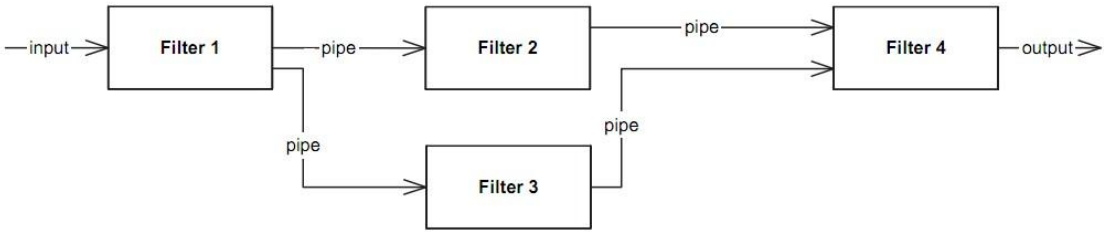

Fig. 12. Pipes and filters example

Shared Repository Pattern:

Data needs to be shared between components. In the shared repository pattern one component of the system is used as a central data store, accessed by all other independent components. This shared repository offers suitable means for accessing the data, for instance, a query API or language. The shared repository must be scalable to meet the clients' requirements, and it must ensure data consistency. It must handle problems of resource contention, for example by locking accessed data. The shared repository might also introduce transaction mechanisms [46].

Active repository:

A system needs to contain a shared repository, but it should not be always so passive. Clients need to be informed of specific events quickly or sometimes even immediately in the shared repository. The shared repository has its limitation in that it does not deliver timely information or inflicts overhead on the system performance.

An active repository is a shared repository that is "active" in the sense that it informs a number of subscribers of specific events that happen in the shared repository. The active repository maintains a registry of clients and informs them through appropriate notification mechanisms.

Blackboard Pattern:

In the case where a shared repository is needed for the shared data of a computation, but no deterministic solution strategies are known. It should be possible to realize a solution for these types of applications.

In a blackboard pattern, the complex task is divided into smaller sub-tasks for which deterministic solutions are known. The blackboard pattern is a specific shared repository that uses the results of its clients for heuristic computation and step-wise improvement of the solution [46]. Each client has the power to access the Blackboard to see whether new inputs are presented for further processing and to deliver results after processing. A control component monitors the blackboard and coordinates the clients according to the state of the blackboard (Fig. 13).

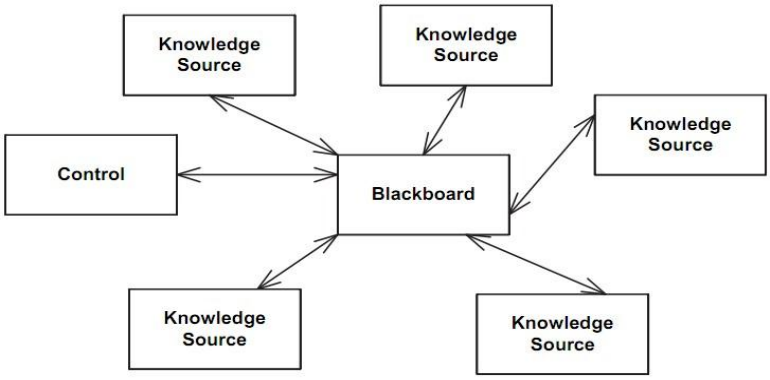

Fig. 13. Blackboard example

Object-oriented pattern:

An object-oriented program will usually contain different types of objects, each type corresponding to a particular kind of complex data to be managed or perhaps to a real-world object or concept such as a bank account, a hockey player, or a bulldozer. A program might well contain multiple copies of each type of object, one for each of the real-world objects the program is dealing with. For instance, there could be one bank account object for each real-world account at a particular bank. Each copy of the bank account object would be alike in the methods it offers for manipulating or reading its data, but the data inside each object would differ reflecting the different history of each account.

An object-oriented program may thus be viewed as a collection of interacting objects, as opposed to the conventional model, in which a program is seen as a list of tasks (subroutines) to perform. In OOP, each object is capable of receiving messages, processing data, and sending messages to other objects. Each object can be viewed as an independent "machine" with a distinct role or responsibility. The actions (or "methods") on these objects are closely associated with the object. For example, OOP data structures tend to "carry their own operators around with them" (or at least "inherit" them from a similar object or class) - except when they have to be serialized [47].

Layered Pattern:

A layered system is organized hierarchically, each layer providing service to the layer above it and serving as a client to the layer below. In some layered systems inner layers are hidden from all except the adjacent outer layer. The connectors are defined by the protocols that determine how the layers will interact [43]. Within each layer all constituent components work at the same level of abstraction and can interact through connectors. In the pure form of the pattern, layers should not be by-passed: higher-level access lower-level layers only through the layer beneath [46].

Layered systems have several desirable properties. First, they support design based on increasing levels of abstraction. This allows people to divide a complex problem into a

sequence of incremental steps. Second, they support enhancement. Because each layer interacts with at most the layers below and above, changes to the function of one layer affect at most two other layers.

Some disadvantages may include hardness of structuring this kind of system and difficulties to find the right level of abstraction.

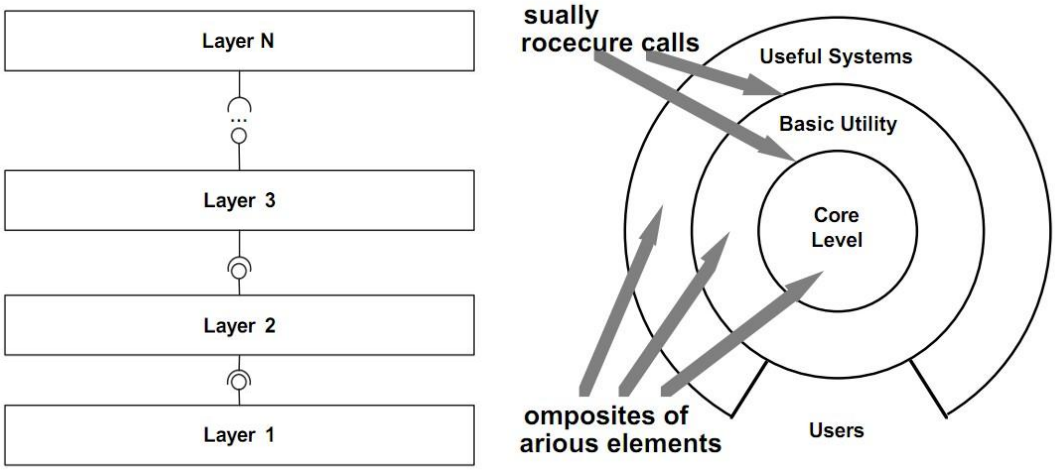

Fig. 14. Layer examples

Program and Subroutine Pattern:

A subroutine may be written so that it expects to obtain one or more data values from the calling program (its parameters or arguments). It may also return a computed value to its caller (its return value), or provide various result values or output parameters. Indeed, a common use of subroutines is to implement mathematical functions, in which the purpose of the subroutine is purely to compute one or more results which values are entirely determined by the parameters passed to the subroutine.

However, a subroutine call may also have side effects, such as modifying data structures in a computer memory, reading from or writing to a peripheral device, creating a file, halting the program or the machine, or even delaying the program's execution for a specified time. A subprogram with side effects may return different results each time it is called, even if it is called with the same arguments. An example is a random number function, available in many languages, that returns a different random-looking number each time it is called. The widespread use of subroutines with side effects is a characteristic of imperative programming languages [48].

Interpreter Pattern:

A language syntax and grammar needs to be parsed and interpreted within an application. The language needs to be interpreted at runtime.

An interpreter for the language is provided, which provides both parsing facilities and an execution environment. The program that needs to be interpreted is provided in form of

scripts which are interpreted at runtime. These scripts are portable to each platform realization of the interpreter.

Rule-based System:

A rule-based system consists mainly of three things: facts, rules, and an engine that acts on them. Rules represent knowledge in form of a condition and associated actions. Facts represent data. A rule-based system applies its rules to the known facts. The actions of a rule might assert new facts, which in turn, trigger other rules [46].

These two pattern catalogues contain the basic patterns we need to analyze and develop our pattern linking framework. Starting from this, in the next chapter, we will discover the internal linking between Business Architecture and Information System Architecture in terms of patterns.

To here, we introduced important information system patterns over. In the next chapter, the information about each pattern will be considered as a reference when we analyze the gap and the possible bridge.

6. Proposals

In this chapter, first, we will have a short review about what we have already discovered. Then we try to give some proposals to bridge the gap.

6.1 Intermediary Summary

From Zachman's enterprise architecture framework and the study of reference architectures, we know that there exists a gap between business architecture and information system architecture in the enterprise architecture. Then we start with studying reference architecture from which we know that the concept of pattern and perspective play an important role in the concrete architectures both in business architecture domain and in information system domain.

In order to give proposals to bridge the gap, some specific business patterns and information system patterns (most in software aspect) and six architecture perspectives are found from references. Now we go back to check the answers for the study questions that we listed before.

- How is Information System Reference Architecture defined?

Answer: This question is answered in Chapter 4.1.

- How is Business Reference Architecture defined?

Answer: This question is answered in Chapter 4.2.

- What are the common parts appearing in the definitions?

Answer: Architecture Pattern and Architecture perspectives.

- What are the architectural perspectives? What are important ones to us?

Answer: This question is stressed in Chapter 5.1.

- What are the patterns in business architecture? How do they function? Why do we use them?

Answer: This question is stressed in Chapter 5.2.

- What are the patterns in information system architecture? How do they function? Why do we use them?

Answer: This question is stressed in Chapter 5.3.

- What is in common appearing in the definitions, which (similar) concepts are being addressed in these definitions and could they be of use to bridge the gap?

Answer: This is part of the main study question and waits to be solved in this chapter.

Now we have only the main study question left. In the section 6.2, we will try to give some proposals.

6.2 Analysis and proposals for possible solutions

To here, it is good to know that pattern and perspectives are discovered as key concepts in bridging the gap between BA and ISA. However, before giving proposals to create a bridge, we have to look deeper in perspectives and patterns to see whether there is a possible solution for the gap.

6.2.1 Difficulties with regard to perspectives

To here, we have found out that there are six perspectives. As to the possibility of bridging the gap, we can have an analysis here.

For aspect perspective, we found that in [10], there is a five aspect framework for IS architecture. The related perspectives are data, organization, system, communication and configuration. However, we didn't find any information about aspects in business architecture. It seems that the aspect framework is only for IS architecture rather than business architecture. A study of aspect in business architecture might help to bridge the gap but we are not so sure of it. Thus, we think that aspect perspective is not a good point to bridge the gap.

For abstraction and aggregation perspective, we can analyze them together because they both describe how much information one architecture could contain. A good thing is that in both business architecture and IS architecture, the two concepts exist. But there is something contradictive here. For one hand, an IS architecture could contain some information in order to let IS architect understand the system; however, if it contains too many details, a business architecture will fail to understand what the IS architecture focuses. For a very complex IS architecture, a business architecture don't understand the terms used and the business architect may even not care of the terms. In order to bridge the gap somewhat, we propose both in BA and ISA, they should contain some information but don't have to be too detailed. The abstraction level and aggregation level should be around 3 which is in the middle of all the levels.

The fourth perspective is Business or Technology orientation. Obviously, business architecture is basically business oriented and IS architecture is technology oriented. This perspective doesn't help a lot in our study because we already know this information. From this perspective, it points out that BA should be closer to technology orientation and ISA should be closer to business orientation in order to bridge the gap. But it still doesn't answer the question of how and what we can do to bridge the gap.

As to the coverage perspective, it points out how many functions an architecture could contain. For example, an Enterprise architecture can only contain an architecture pattern (MVC, for example), or an Enterprise architecture contain all the information in terms of patterns, reference models covering both business and IT aspects. We think that this perspective is not helpful in bridging the gap. This is due to the fact that whether a IS

architecture contains only pattern or contains reference model doesn't enhance the understanding for a business architect. More coverage can't make sure the understandability of an ISA for a business architect. Besides, we found that the coverage concept is properly used in Enterprise Architectures that contain both business and IT aspects but is not used solely on either of them.

As to the stakeholder perspective, we can do something useful. There are stakeholders in both kinds of architectures. For example, for a BA, the related stakeholder could be a manger or customer and for an ISA, the related stakeholder could be a system engineer. Every stakeholder has his or her background and knowledge. Here we have a proposal that according to each stakeholder with their concerns, a BA or an ISA could be changed into another form which that the corresponding stakeholder could understand easier and get more information. However, a difficulty of doing that needs more costs in terms of times and people etc. Due to the limited time in our study, we don't prepare to do it here but we propose a future study on this direction.

In summary, for an overall conclusion, architecture perspective is not a good choice to bridge the gap. Some of them are helpful, such as stakeholder perspective, but the help is still limited. Some of them even have no help, such as Business or IT orientation. To here, we decide to exclude more study on architectural perspectives and we turn to patterns. A good thing is that when we focus on patterns, we found that the concept of perspectives still appears and makes contribution to bridge the gap.

6.2.2 Difficulties with regard to patterns

There are several difficulties when we try to directly build connections between patterns found in BA and ISA. When we examine the business patterns, we found that many of them are too close to business functions which mean it is hard to interrelate directly to IS side. For example, the commitment pattern solves the problem that how we model promises of future economic events. Contract pattern defines what would happen if the commitments are not satisfied. Location pattern solves the problem how we specify the place where the economic events occur. In these business patterns, no information system concept is mentioned. As to the IS pattern we have found, we also found that these patterns are too close to technical aspect, more specifically, software techniques. For example, the pipes and filter pattern and batch sequential pattern provides a software development logic. The implicit and explicit invocation pattern defines when new programs are invoked. These patterns are too technical that no business concepts are reflected.

Although both in BA and ISA, patterns appear; however, it seems that the meaning of this term is not totally the same. They use the recurring property of a pattern but there is

still a big difference existing because one is business-oriented and another is technology-oriented.

In summary, although pattern is a common concept; however, the gap between business pattern and information system pattern is still very big which makes it hard or even impossible to build any connections directly between them. The interrelation in patterns is small. Even though that it is hard to bridge the gap directly; however, we can still attempt to make some efforts towards bridging the gap. In section 6.2.2, several proposals are made.

6.2.3 Proposals towards bridging the gap

Although the gap is rather big; however, we can make some proposals to the current situation to bridge a gap somewhat.

Proposal 1: Build a new classification of IS patterns

When we look at the classifications of business pattern and IS pattern in figure 9 and 11, we found that the structures are rather different. In business pattern classification, basically, the patterns are classified into two groups, structural and behavior patterns. In IS patterns, there is no such classification. A reclassification of IS pattern is helpful because more similarities provide more possibilities to bridge the gap between two architectures. Since business patterns have structural and behavioral classifications and if IS patterns have that classification, it is possible to discover further relationships in the two directions. In fact, in software world, classification of design style has already been discriminated into behavioral and structural orientations [57], so it is not a new concept in technical world. However, there is no clear distinction in the pattern aspect.

Before classifying the IS patterns in figure 11 into behavioral and structural orientation, the corresponding criteria should be made. In [58], behavior is described as actions and mannerisms made by organisms and systems to solve a problem. In [59], a structural pattern should help identify and describe relationships between entities. We can take these two rules as a criterion and if a pattern is closer to one of them, we can classify it as the closer group. According to this, a proposal of classification for IS patterns can be obtained in terms of behavioral and structural dimensions.

In the previous introduction, we give introduction of both implicit invocation pattern and explicit invocation pattern. Implicit invocation pattern and explicit invocation pattern provides a mechanism to invoke the related component, like supplier or client. They use different ways to tell other components when tasks appear. They are designed for the same goal to invoke related components. According to the classification criterion, these two patterns are closer to behavior patterns since they solve the problem of how to do something.

Pipes & Filters pattern and Batch Sequential pattern should belong to structural patterns according to the criterion. This is due to the fact that in pipes & filters, the system is designed as a logical structure; a chain of processes where the output of each process is the input of the next. Batch sequential pattern is the same as Pipes & filters to some extent but provides different sequential data processing logic.

Repository pattern and blackboard pattern should belong to structural patterns. We say this because these two patterns focus on how the data is shared, whether through a data repository or a more heuristic blackboard way. They tell a data centered structure which indicates that they should belong to structural patterns. We can also know this from the above classification schema that they both under category of Data Centered Pattern.

For object-oriented pattern, layered pattern and program subroutine pattern, we believe that they should belong to structural patterns. They point out how the program is organized and a structure about how to treat objects. More obviously, the layered pattern suggests a layered structure.

For Interpreter pattern and Rule-based system, we believe that they should belong to behavioral patterns. They point out how the system reacts according to the corresponding pattern. One is through interpretation and another one is based on rules. System acts differently with different mechanisms.

In the next table, a table is summarized. (Table 3)

Pattern Group	Information System Pattern	Description
Structural Pattern	Pipes & Filters	Roughly speaking, these information system patterns are more likely to give a structural recurring logic.
	Batch Sequential	
	Repository	
	Blackboard	
	Object-Oriented	
	Layered	
	Program subroutine	
Behavioral Pattern	Explicit Invocation	Roughly speaking, these information system patterns are more likely to describe system behaviors.
	Implicit Invocation	
	Rule-based System	
	Interpreter	

Table 3. IS Patterns classified into Structural and Behavioral Groups

Through doing this, the gap is somewhat be bridged in the sense that both ISA and BA's patterns can be classified into behavioral and structural group. Although it is still hard to link them directly; however, this similarity gives us a hint of a study direction of the two groups.

Proposal 2: Insert a new kind of pattern

Since that the gap is so large that we don't have to link them directly between IS patterns and business patterns. Imagine that business pattern is on one side of a river and IS pattern is on the other side. A new kind of pattern can be inserted as an intermediary in the middle of the river. It is near to both IS and business patterns and contains both business terms and IS terms. It links both business patterns and IS patterns. Here, we name it Business Information System (BIS) patterns. As we have said, a business pattern is very close to business functions. The BIS pattern has to interpret it so that they can be understood by IS somewhat. However, a singer interpreter can help understand better of the business patterns but doesn't help too much to link to the IS patterns. Then transformer is needed to serve for the linking. We get the hint of interpreter and transformer logic from [61], in which that this logic functions well in bridging the gap between different programming languages. According to this information, BIS patterns should at least contain the following characteristics.

1. An interface to business pattern
2. Interpreters to interpret business pattern in different perspectives.
3. Transformers
4. Interfaces to IS patterns

An interface is needed to link business patterns. It doesn't have to be very complex but it is important as a starting point bridging the gap. Besides, there should be several interpreters as well. For a given business pattern, who are the stakeholders, what data aspect is important? To answer these questions, interpreters are needed. Then having obtained the information from interpreters, transformers play an important role to transform the interpreted result into IS domain. Then with the help of interfaces to IS patterns, the IS patterns are finally connected. A figure showing the structure of BIS patterns is given as follow:

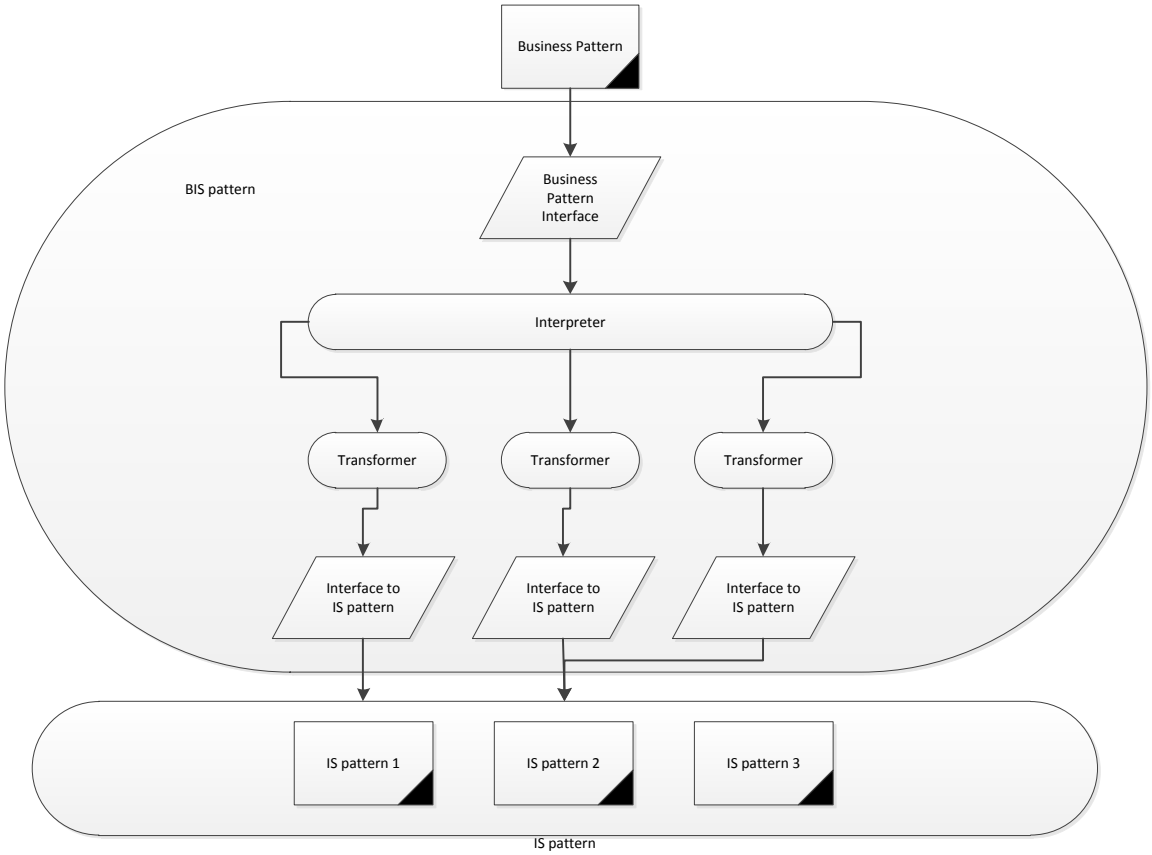

Fig. 15. BIS pattern

This is a proposal of a framework for BIS patterns. It is possible that with further study, this framework can contain more elements. In our research, we aim at giving a rudiment of a structure to bridge the gap somehow. In the next part, we try to give some examples to show how this BIS pattern could be like.

(A) Contract Pattern

The first example is contract business pattern. Before building connections, we should look further into this business pattern. Basically, A contract is [60]:

a) an agreement between two or more parties, especially one that is written and enforceable by law, or

b) the writing or document containing such an agreement.

The structure of the contract pattern is illustrated in the next figure.

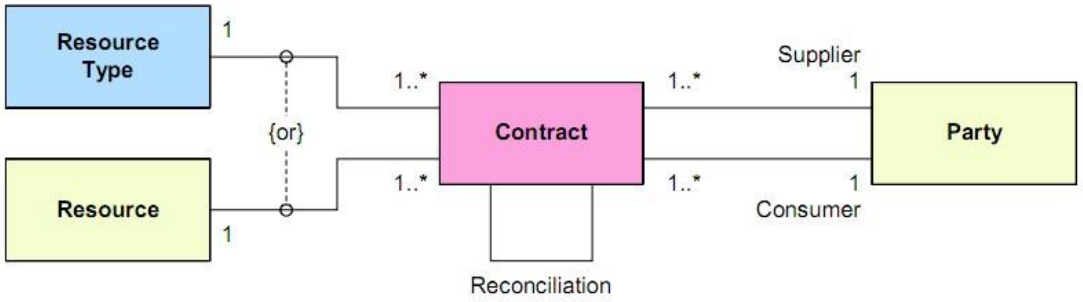

Fig. 16. Structure of the contract pattern [60]

The relationship between parties is encapsulated in the entity called contract. The contract entity is related to two party entities: the supplier and the consumer. Examples of parties are customer, employee etc. Each party represents a physical person or organization. Each contract is related to one resource or resource type. Each resource or resource type represents an asset, a physical product or other subject of trade.

Several forces have to be considered:

1. Parties have relationships between each other. We can model these relationships as associations between the parties. However, these relationships often have specific attributes, describing details of this relationship rather than details of one of the parties. Examples of such attributes are delivery due, validity period, payment terms and employment position.
2. Relationships between parties involved in business are manifested in various documents, such as purchase and sales orders, quotes, invoices, payments, and delivery receipts.
3. If a relationship exists between parties, this relationship may, under conditions, create or cause the creation of other relationships between the same parties. For example, a purchase order may result in a delivery of goods.

Given such a contract pattern in business architecture, an interface in BIS pattern should be connected to the contract pattern as mentioned above. Through the interface, the basic attributes are captured. Then an interpreter is used to analyze the pattern in different perspectives, such as aspects, coverage and stakeholders. Then according to the results of the interpreter, each of them would be transformed into related IS patterns and connected with IS patterns through interface. The next picture gives a better view of contract BIS pattern:

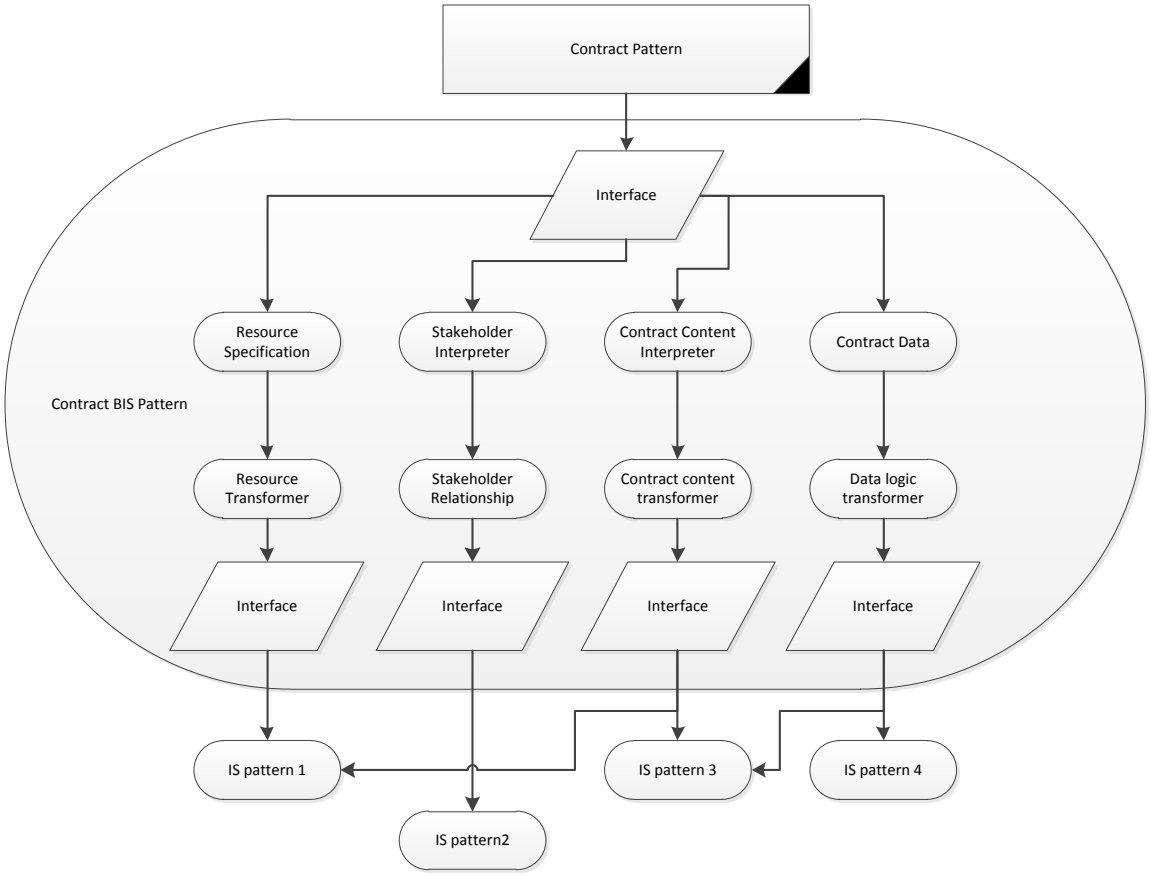

Fig. 17. Contract BIS pattern [60]

In this BIS pattern, it is connected with contract pattern in business architecture. Then according to the contract pattern structure showed above, there are several important aspects, such as stakeholder perspective, contract content, resource type and contract data. With the help of corresponding interpreter, more information is obtained. For example, what is the logic behind a piece of content in the contract? Is it behavioral or structural? As to data interpreter, the data logic is concluded like how much money involved and how they are transferred and when is the delivery time. After that, the transformer gets the information and acts effectively. For contract content: The supplier has to deliver the goods within 15 days after the customer pays the goods. Given this, the contract content transformer would come up with a result:

(1) If the customer pays, the supplier knows that and is informed to deliver.

(2) The 15 days needed to be counted.

To here, it is easier to connect IS patterns with the above functions. For (1), the transformer would build a bridge that link the implicit invocation pattern to the logic. For (2), a counter or timer pattern in IS could be proper. After that, with the help of IS interface, the connection to the target IS pattern(s) is built.

We don’t know what the interpreter, transformer and interfaces are really like in business information system applications. For every business pattern, the corresponding BIS pattern should have its specific structures and functions. In contract BIS pattern, the structure is based on contract pattern in business architecture.

(B) Schedule pattern

Another example is schedule pattern. Schedule is a series of things to be done or of events to occur at or during a particular time or period. [37]

If use, consume, and other economic events do not happen as commitments predefined, the enterprise would like to have an alternative scheme to mitigate the consequences. Application developers would like to integrate this function in enterprise applications. More specifically, in a schedule pattern, a stakeholder is scheduled to do something at specific time. If he finishes the job in time, then he comes to the next task. If he can’t finish the task, the predefined alternative task can be considered to mitigate the possible impact of no fulfillment. A good schedule not only restricts in scheduling, but also considers carefully about the exceptional handling. A simple example of schedule pattern is as follows:

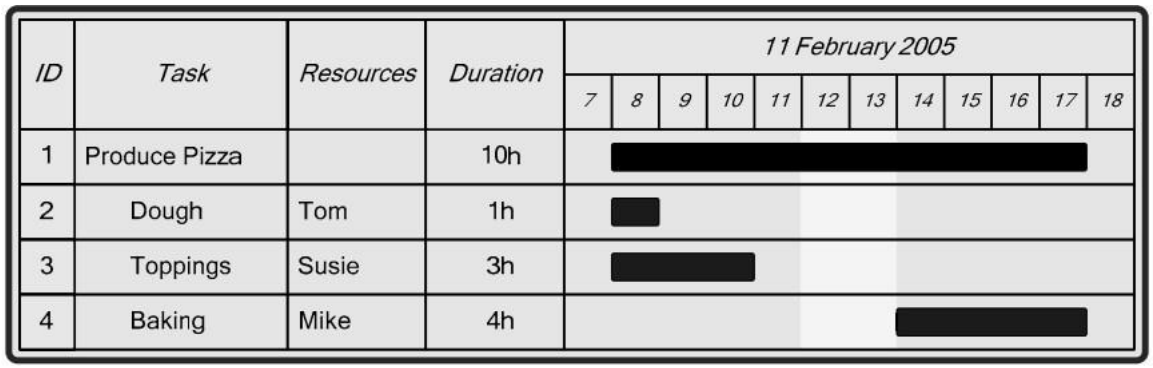

Fig. 18. A simple schedule example

The following forces have to be considered:

1. If use, consume and produce economic events don’t occur as commitments specify, the enterprise would like to have an alternative plan to mitigate the consequences. Application developers would like this information present in the business application.
2. A conversion process usually consists of several use, consume, and produce economic events that have various, often complex dependencies on each other. If some of these events do not occur as committed, the mitigation play depends on a combination of the values of the economic events.

Just like what we have proposed for contract pattern, we can also propose a BIS pattern for schedule pattern. However, we should know that it differs from contract pattern and the corresponding interpreters and transformers are different (Fig 18).

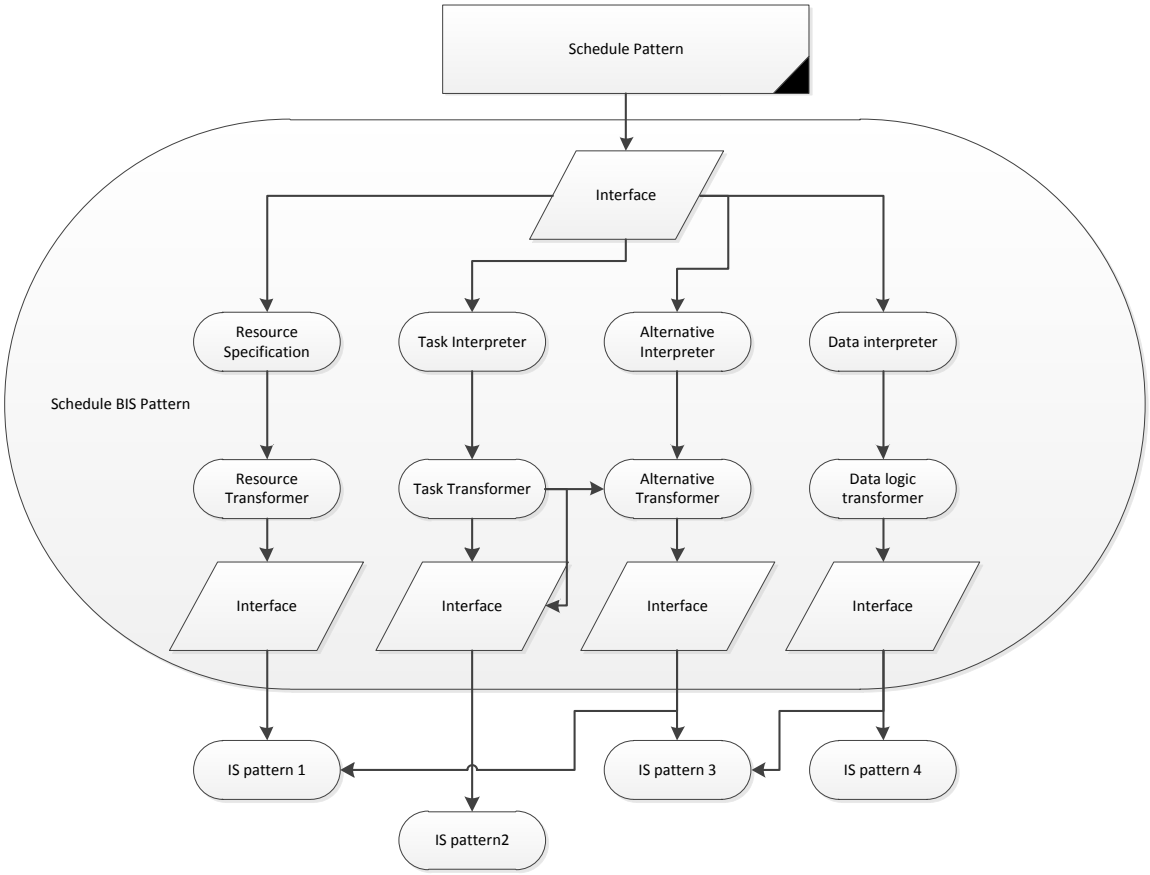

Fig. 19. Schedule BIS pattern [60]

After the business pattern interface, the schedule pattern could be classified in four aspects, resource, task, alternative and data aspect. The resource and data aspects are similar with contract pattern. The corresponding interpreters and transformers can be a bridge to link to IS patterns. However, for task transformer and alternative transformer, on one hand, they have their own logics; on the other hand, if original task is default, the alternative should be invoked. This logic also needs information system patterns to realize. As a result, it needs more interfaces and IS patterns related. For example, in a schedule pattern:

(1) Original task: A should keep on working continuously for 15 days. If A doesn't, execute alternative task.

(2) Alternative: B should continue the work left by A.

For both original task and alternative task, they have their own logic in terms of resources and times. Except that, an invoker mechanism is needed when A fails. This mechanism need helps from IS patterns and should be connected to IS patterns through interfaces.

From these two examples, we can have an idea about how the BIS pattern could be like. The BIS patterns, serve as an intermediary to connect business patterns and IS patterns so that the gap between business architecture and IS architecture could be bridged somewhat.

If designers have difficulties to understand the relationship between the two kinds of architectures, they may look BIS patterns for help.

Proposal 3: Insert a new kind of Architecture

Look back at our study process. We start from Zachman's work and we know that in Enterprise Architecture, the gap between business architecture and IS architecture exists. Then we found an important concept in them two. The concept is pattern. But we found that even though pattern is a common concept, the gap is still too big to connect business architecture and IS architecture directly. Then in the last chapter, we propose to insert another type of patterns as an intermediary. We name it as BIS pattern. We also know how the business reference architecture and IS reference architecture is defined in the first part of this report. To here, we are considering a new question. Is it possible to add another kind of architecture not only contains patterns concept but also other elements somewhere between BA and ISA like the one in figure 19?

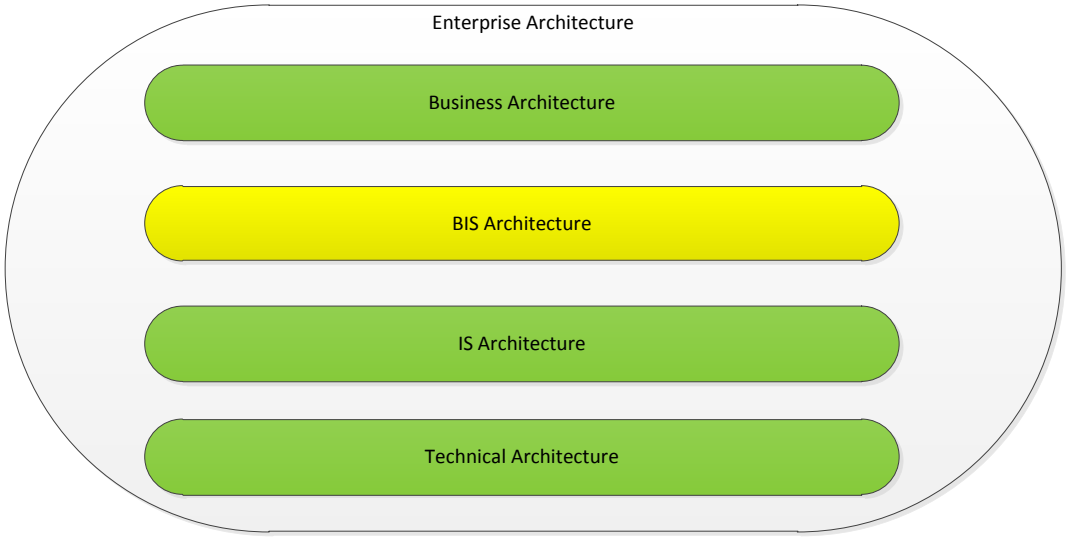

Fig 20. A new structure of Enterprise Architecture (Modified from figure 1)

To define the BIS architecture properties, similarly, we can try to define BIS RA. In BIS RA, BIS pattern which we discussed above is an important concept. As to the other concepts, we are not sure yet. It is possible that there is "BIS Reference Model" just like business reference model we discovered in business RA. We propose that the aim of BIS architecture is a bridge to shorten the gap between BA and ISA. Due to the time limitation and topic in our study (see title), we propose a future study on details of BIS architecture.

7 Conclusion

Our goal of research is to try to bridge the gap between business architecture and information system architecture in the enterprise architecture. In the beginning, how these two kinds of architectures are defined is studied. Then we could know what concepts are important concepts in them. Based on that, we can find the key concepts of connecting them. The key concepts include architecture perspectives and patterns. Then we did deeper research to explore more information in architecture perspectives and find some specific perspectives. However, after series of analysis, we don't think that architecture perspective is a good direction overall. But some proposals can be made based on some of the perspectives such as stakeholder and aspect perspectives. Then, we discovered how business patterns and information system patterns are defined and explain more on some of them. We didn't list all business and information system patterns because there are thousands of them in every business area. Through further analysis, we found that the gap is still very big that a direct linkage between IS pattern and business pattern is hard. Given this result, we give several proposals. First one is to have a new classification of IS patterns in behavioral and structural distinctions. Through doing is, IS patterns have one more similarity with business patterns. Both of the two kinds of patterns can have two groups, structural and behavioral. Another proposal is to insert an intermediary kind of pattern that helps to bridge the gap. We call it BIS pattern. In one hand, it connects business patterns. On the other hand, it connects IS patterns. Through doing this, we believe that the gap between IS architecture and business architecture is somewhat bridged in a pattern language. A third proposal is to insert another type of architecture between BA and ISA. We can call it BIS architecture containing the BIS patterns we propose above and some other elements. However, due to the limitation of time and our study topic, we don't explore too many details of it.

Future work is needed to make the gap better bridged. First of all, studies could still be done in directly bridging the gap between business patterns and IS patterns especially after we propose a new classification for IS patterns. Besides, we also propose more studies on the possibility of bridging the gap in terms of architecture perspectives, for example, the stakeholder perspective. Furthermore, the BIS pattern and BIS architecture inserted is also an interesting topic that more studies could be made on this concept. We hope that the gap between BA and ISA could be bridged or narrowed successfully in the near future.

8. Acknowledgement

All members in assessment committee, including Mr. Trienekens, Mr. Kusters and Mr. Somers, are thanked for their comments of the research and report. Special thanks for my supervisor Mr. Trienekens for his long time support during the whole research. Mrs. Sanders is thanked for her advice on arranging research schedule.

9. Reference

[1] Pereira, C. M. and P. Sousa (2004). A method to define an Enterprise Architecture using the Zachman Framework, ACM.

[2] Zachman, J. (2009). "The zachman framework: the official concise definition." Available on http://www. zachmaninternational. com, Accessed 16.

[3] http://www.eacommunity.com/resources/download/bolton_what.pdf

[4] Chalmeta, R., C. Campos, et al. (2001). "References architectures for enterprise integration." Journal of Systems and Software 57(3): 175-191.

[5] http://en.wikipedia.org/wiki/Reference_architecture

[6] Chen, D., B. Vallespir, et al. (1997). "GRAI integrated methodology and its mapping onto generic enterprise reference architecture and methodology." Computers in industry 33(2-3): 387-394.

[7] http://www.boozallen.com/media/file/IT_EnterpriseArchIntegration_FS.pdf

[8] Kitchenham, B. (2004). "Procedures for performing systematic reviews." Keele, UK, Keele University 33: 2004.

[9] Budgen, D. and P. Brereton (2006). Performing systematic literature reviews in software engineering, ACM.

[10] Grefen, P. (2012). Business Information System Architecture, Eindhoven University of Technology.

[11] Samuil Angelov, J. J. M. T., Paul Grefen (2011). Extending and Adapting the Architecture Tradeoff Analysis Method for the Evaluation of Software Reference Architectures. Eindhoven, Eindhoven University of Technology.

[12] Cloutier, R., G. Muller, et al. (2010). "The concept of Reference Architectures." Systems Engineering 13(1): 14-27.

[13] Muller, G. (July 1, 2011). "A Reference Architecture Primer." Embedded Systems Institute.

[14] Estefan, J. A., K. Laskey, et al. (2008). "Reference Architecture for service oriented architecture version 1.0." Public Review Draft 1.

[15]Wikipedia."ReferenceArchitecture."http://en.wikipedia.org/wiki/Reference_architecture.

[16] Gallagher, B. P. (2000). Using the Architecture Tradeoff Analysis MethodSM to Evaluate a Reference Architecture: A Case Study, DTIC Document.

[17] Antunes, G., J. Barateiro, et al. (2010). "A reference architecture for digital preservation." Proc. iPRES2010.

[18] Versteeg, G. and H. Bouwman (2006). "Business architecture: A new paradigm to relate business strategy to ICT." Information Systems Frontiers 8(2): 91-102.

[19] Wolfenden, P. J. and D. E. Welch (2000). "Business architecture: a holistic approach to defining the organization necessary to deliver a strategy." Knowledge and Process Management 7(2): 97-106.
[20]
http://en.wikipedia.org/wiki/Business_architecture#Different_views_of_an_organization
[21] McDavid, D. W. (1999). "A standard for business architecture description." IBM systems journa`l 38(1): 12-31.
[22] http://pubs.opengroup.org/architecture/togaf9-doc/arch/chap08.html
[23] http://www.immi.gov.au/managing-australias-borders/border-security/systems/ibrm-executive-report.pdf
[24] Wieringa, R., H. Blanken, et al. (2003). Aligning application architecture to the business context, Springer.
[25] Fattah, A. (2009). Enterprise Reference Architecture.
[26] J. Truijens, A. Oosterhaven, R. Maes, H. Jägers, F. van Iersel; Informatieinfrastructuur: een Instrument voor het Management; Kluwer Bedrijfs-wetenschappen, 1990.
[27] Collins, J., J. Greer, et al. (1996). Adaptive assessment using granularity hierarchies and Bayesian nets, Springer.
[28] Wu, H. (2002). A RA for Adaptive Hypermedia Applications, Citeseer.
[29] Grefen, P. (2010). Mastering e-business, Routledge.
[30] Arsanjani, A., L. J. Zhang, et al. (2007). "S3: A service-oriented Reference Architecture." IT professional 9(3): 10-17.
[31] Bernus, P. and L. Nemes (1996). "A framework to define a generic enterprise Reference Architecture and methodology." Computer Integrated Manufacturing Systems 9(3): 179-191.
[32] Barber, K. S., T. J. Graser, et al. (2001). A multi-level software architecture metamodel to support the capture and evaluation of stakeholder concerns, Citeseer.
[33] Antunes, G., J. Barateiro, et al. (2010). "A reference architecture for digital preservation." Proc. iPRES2010.
[34] Proper, H.A., Verrijn-Stuart, A.A., Hoppenbrouwers, S.J.B.A. (2005)."On Utility-based Selection of Architecture-Modelling Concepts", In Proceeding of the 2nd Asia-Pacific conference on Conceptual Modelling, Newcastle, NSW, Australia.
[35] http://it.toolbox.com/blogs/enterprise-design/defining-business-patterns-4704
[36] http://www.users.globalnet.co.uk/~rxv/cbb/cbb-businesspatterns.pdf
[37] Hrubý, P., J. Kiehn, et al. (2006). Model-driven design using business patterns, Springer.
[38] http://www.gofpatterns.com/design-patterns/module5/structural-design-pattern.php

[39] http://en.wikipedia.org/wiki/Behavioral_pattern
[40] Gamma, E., R. Helm, et al. (1993). "Design patterns: Abstraction and reuse of object-oriented design." ECOOP'93—Object-Oriented Programming: 406-431.
[41] Bass, L., P. Clements, et al. (2003). Software architecture in practice, Addison-Wesley Longman Publishing Co., Inc.
[42] http://en.wikipedia.org/wiki/Implicit_invocation
[43] Garlan, D. and M. Shaw (1993). "An introduction to software architecture." Advances in software engineering and knowledge engineering 1: 1-40.
[44] S. P. Reiss, "Connecting tools using message passing in the field program development environment," IEEE software, July 1990.
[45] K. Sullivan and D. Notkin, "Reconciling environment integration and component independence," in Proceeding of ACM SIGSOFT90: Fourth Symposium on Software Development Environments, pp. 22-23, December 1990.
[46] Avgeriou, P. and U. Zdun (2005). "Architectural Patterns Revisited–A Pattern Language."
[47] http://en.wikipedia.org/wiki/Object-oriented_programming
[48] http://en.wikipedia.org/wiki/Subroutine
[49] Ambler, Scott W., Agile Enterprise Architecture: Beyond Enterprise Data Modeling, 2002, http://www.agiledata.org/essays/enterpriseArchitecture.html
[50] http://www.technical-communicators.com/articles/zachman_framework.pdf
[51] McDavid, D. (2003). "The business-IT gap: A key challenge." IBM Research Memo, http://www. almaden. ibm. com/coevolution/pdf/mcdavid. pdf.
[52] Hollingsworth, D. (1995). "Workflow management coalition: The workflow reference model." Document Number TC00-1003(1.1).
[53] De Bra, P., G. J. Houben, et al. (1999). AHAM: a Dexter-based reference model for adaptive hypermedia, ACM.
[54]http://www.cs.uwaterloo.ca/~a78khan/cs446/lectures/2011_05may_27_ArchitecturalTypes.pdf
[55] Veeramani, D., B. Bhargava, et al. (1993). "Information system architecture for heterarchial control of large FMSs." Computer Integrated Manufacturing Systems 6(2): 76-92.
[56] Veeramani, D., B. Bhargava, et al. (1993). "Information system architecture for heterarchial control of large FMSs." Computer Integrated Manufacturing Systems 6(2): 76-92.
[57] Rugaber, S. and S. Vattam (2009). "Structure, behavior, and function of complex systems: the structure, behavior, and function modeling language." Artificial Intelligence for Engineering Design, Analysis and Manufacturing 23(1): 23-35.

[58] http://en.wikipedia.org/wiki/Behavior
[59]http://www.developintelligence.com/sites/default/files/assets/learn/structural-patterns.pdf
[60] Hruby, P. and F. Allé"Contracts A General Pattern for Business Modeling."
[61]http://www.csd.uwo.ca/~watt/cs4447/notes/CS4447%2001%20--%20Interpreters%20and%20Transformers.pdf