Seamless design of information system architecture based on adaptive clustering method

Grigory Tsiperman
Assistant professor
National University of Science and Technology "MISIS"
gntsip@gmail.com

Abstract. The paper considers the concept of building the architecture of an information system that provides a seamless connection between architectural representations of various levels of abstraction. The concept is based on the application of the adaptive clustering method of information systems developed by the author. Seamless connection is understood as the presence of connections between elements of architectural models related to architectural representations of various levels of abstraction.

Keywords: information system, architectural abstraction, design, architectural model, seamless link, technological gap, adaptive clustering method

1 Introduction

This paper considers consider the issues of building an information system (IS) architecture in the context of enterprise architecture. The concept of IS architecture historically preceded the concept of enterprise architecture, which arose from the realization that the IS model must meet the requirements of the business and be able to flexibly adapt to its needs. Moreover, such compliance should not be fragmented, it should be based on a deep understanding of the business and its development prospects. IS should ensure the implementation of business requirements, have the ability to adapt to its changes.

The idea of enterprise architecture is to create interconnected architectural models that combine the concepts of mission, goals, enterprise business strategy, business processes, information systems, etc. To implement the idea of enterprise architecture, several methodologies were proposed (see, for example, the reviews in [1] and [2]), none of which are without drawbacks and to this day is not a paradigm.

What all the methodologies agree on is the need for the concept of the life cycle of an enterprise and, accordingly, IS. For the TOGAF [3], GERAM [4] and FEA [5] ICs, the identical stages of the life cycle and the types of IS architecture corresponding to these stages are determined:

- Business architecture
- IS architectures (data architecture and application architecture)
- Architecture of technology.

This paper discusses the construction of IS architecture, so the IS model under consideration includes only the necessary aspects of the enterprise business process model. Moreover, the work does not concern the construction of a basic (as-is) model, confining itself to the issues of constructing a target (to-be) architecture of IS based on a target business model.

One way or another, starting with Zakhman [6], the methodologies for creating IS architecture suggest a downward development, with the consistent construction of architectural models of an ever-increasing level of detail. The levels of detail correspond to certain stages of the IS life cycle: the concept is created by presenting the target business architecture, technical design involves the description of functional architecture, component architecture, data architecture, and determines the technology architecture of IS.

The practice of designing IS architecture involves a heuristic transition between architectural models of various levels of abstraction: a more detailed architecture is constructed in such a way that the technical solutions match the models of the previous level as much as possible. In this case, the architect uses his creativity, experience and knowledge to build a detailed model, and then proves (or considers it obvious) that the resulting model satisfies the requirements arising from a more abstract architecture. In the heuristic approach, the relationship between architectural models of various levels of

abstraction is inverse evidence-based. Many developers paid attention to this aspect of IS design (see [7], [8], [9]). Let us designate it as a technological gap between architectural models of various levels of abstraction.

In fact, we are talking about verification and validation[1] the architectural models that underlie the program code. Technological gaps in the absence of verification and validation processes can lead to costly errors.

Unlike the heuristic transition between architectural models of various levels of abstraction, which creates technological gaps, this article considers a smooth transition in which such gaps do not arise. The technology under consideration relates to the development of functional IS architectures. It is based on the application of the adaptive clustering method (ACM, see [10], [11]), in which detailed architectural models are justified by high-level abstraction models, ensuring their seamless connection and the possibility of tracing between components of architectural models.

2 Formulation of the problem

The architectural description of the system is a set of architectural representations corresponding to various points of view on the system. Consider Figure 1, representing a fragment of a conceptual model of architectural representation [12], supplemented by an architectural element that defines a seamless connection between architectural representations.

The architectural representation corresponds to a certain point of view on architecture, and defines the architecture of IS with a degree of detail corresponding to the stakeholder. In Zakhman's scheme, for example, the points of view correspond to the participants in the process of creating the system: a planner, analyst, architect, designer, programmer and operator. Considered in the indicated sequence, these points of view lead to associated architectural representations of an ever-increasing level of detail. An architectural representation includes one or more architectural models that reflect the relationship of the elements of the architecture description (hereinafter referred to as the elements). Elements correspond to abstracts determined by the type of model and the point of view on the architecture of IS.

The main objective of this work is to identify among the elements of architectural models included in the architectural representation of the corresponding level of abstraction, an element (connecting element), the decomposition of which defines the elements of architectural models of the next level of abstraction.

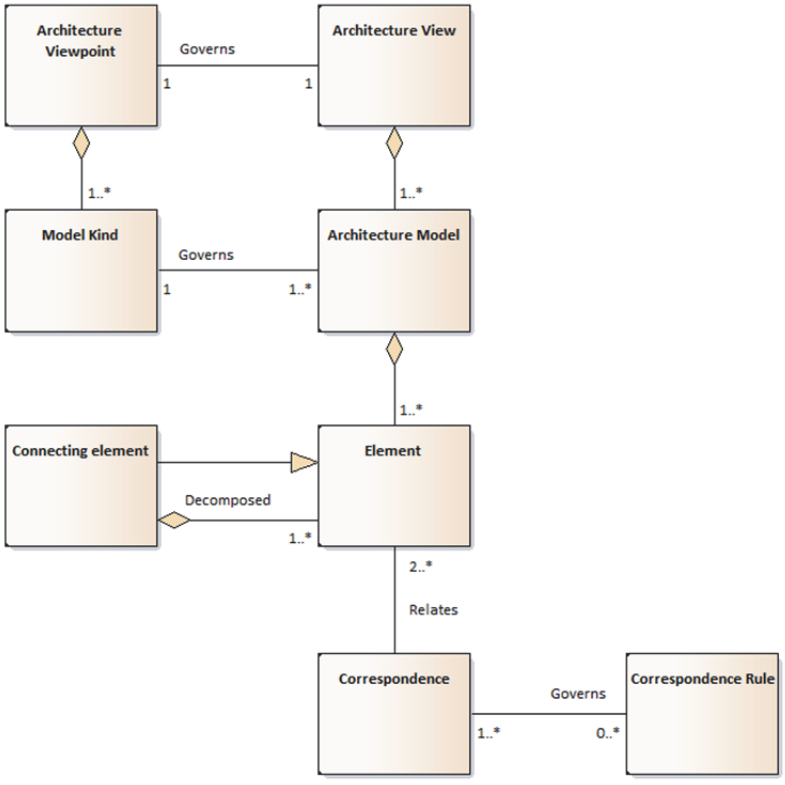

Figure 1 – Conceptual model of architectural representation

The connecting element clearly defines the relationship between the elements of architectural models of various architectural representations. The essence of the technological gap between architectural concepts is that with a heuristic approach to design, the relationships between the elements of architectural models are not defined explicitly, so the validation and verification of architectural models is a rather complicated, painstaking task.

[1] Validation is a confirmation (based on the presentation of objective evidence) that the requirements intended for a particular use or application are fulfilled, and verification is a confirmation that the specified requirements are fulfilled [15].

By seamless connection, we mean the presence of explicit connections between the elements of architectural models of various architectural representations. In this case, the description of the IS architecture is an interconnected set of architectural representations, and for elements of architectural models, tracing[2] becomes possible, which makes the process of model validation and verification trivial - you just need to make sure that the decomposition of the connecting elements is performed correctly.

In this paper, based on the definition of connection elements, describe a seamless connection scheme of architectural representations of the target business architecture, functional architecture, component architectures, data architecture and deploying architecture IS.

3 Business Architecture Design

At ACM, it is business architecture that is the starting point for IS design. In the context of IS design, business architecture is understood to mean many models of automated business processes of an enterprise. In the context of this work, we consider the target business architecture, i.e. business process models taking into account the use of IS. The task is to determine the complete set of automated functions of the business process.

3.1 Business Architecture Concepts

The architectural models of business architecture are built on three main elements (Figure 2), which are defined in the decomposition procedure:

- a business process, understood in the usual sense, as an action in which, based on one or more types of input objects [resources], a result valuable to the client is created; a business process can be represented by a set of actions determined on the basis of its decomposition;
- a business function is an action from a set of actions determined by the decomposition of a business process;
- a business operation is a business function that cannot be decomposed, the executor of which is a specific employee.

Decomposition determines the explicit relationships between these architectural elements: a business process is decomposed into business functions, business functions, in turn, are decomposed into business functions and business operations. The process ends when all business functions are decomposed into business operations.

A business operation is a place where a user interacts with an IS, and accordingly, explicitly or implicitly, includes automated functions that support it.

3.2 Operational Service

The business architecture defines the functional requirements of the user for IS. Description of business operations allows you to define functions that should be automated. The set of these functions, defined for a business operation, constitutes the content of the business operation service (hereinafter referred to as the operational service). Operational services are formed for each business operation and include automated functions descriptions. They are architectural elements of an abstract nature that do not imply any implementation.

An operational service is a connecting element that defines a seamless relationship between business architecture and functional architecture.

Figure 2 demonstrates this relationship: the operational service is decomposed into system dialogs, which are the main element of the functional architecture that provides access to IS functions. Automated functions of a business operation are implemented in dialogs.

In addition, the operational service is the boundary between the business architecture and the system architecture. It formalizes the requirements for automating business operations and serves as the basis for the formation of software requirements specifications.

4 Functional Architecture

Functional architecture is understood as an architectural representation, including architectural models of the structure and composition of the functional components of IS, providing access to the "internal" functions of IS that implement automated functions. In other words, the functional architecture models the interaction of IS with users, as well as with other external agents. The functional architecture forms the appearance of the IS based on the presentation of the compositions of the dialogs of the system, defines the requirements for dialogs and specifies interfaces with external systems.

A dialogue is any act of agent interaction that causes a change in the state of the IS by launching the corresponding software components [9]. Thus, dialogue is understood in a broad sense - this is not only the interaction of the user with the computer, but also the exchange of messages between any IS objects, as well as external agents.

The structure of the dialogue description (Figure 2) includes the following architectural elements:

[2] The relationship between the elements of architectural models different levels of abstraction.

– The function of the presentation level of the IS (view function) is the IS function implemented in the dialogue, providing access to the "internal" IS functions. In other words, through the view functions, the user interacts with the IS.
– The source resource and the Target product are information objects, respectively, received at the input of the dialogue upon its initiation and formed as a result of the dialogue at the exit from it.
– Dialog form - representing a dialog for the user, if the user is supposed to be an agent.

The view functions defined in the dialog have the following classification:

– implementation of dialogue preconditions upon initiation;
– input / output of data field values;
– processing of dialogue control elements that change the state of the IS;
– reaction to errors and contingencies during the dialogue;
– implementation of postconditions at the end of the dialogue.

Functional architecture defines the next level of architectural representation - component architecture. A connecting element that defines seamless connections between functional architecture and component architecture are view functions that are decomposed into software modules. A detailed discussion of component architecture is provided in the next section.

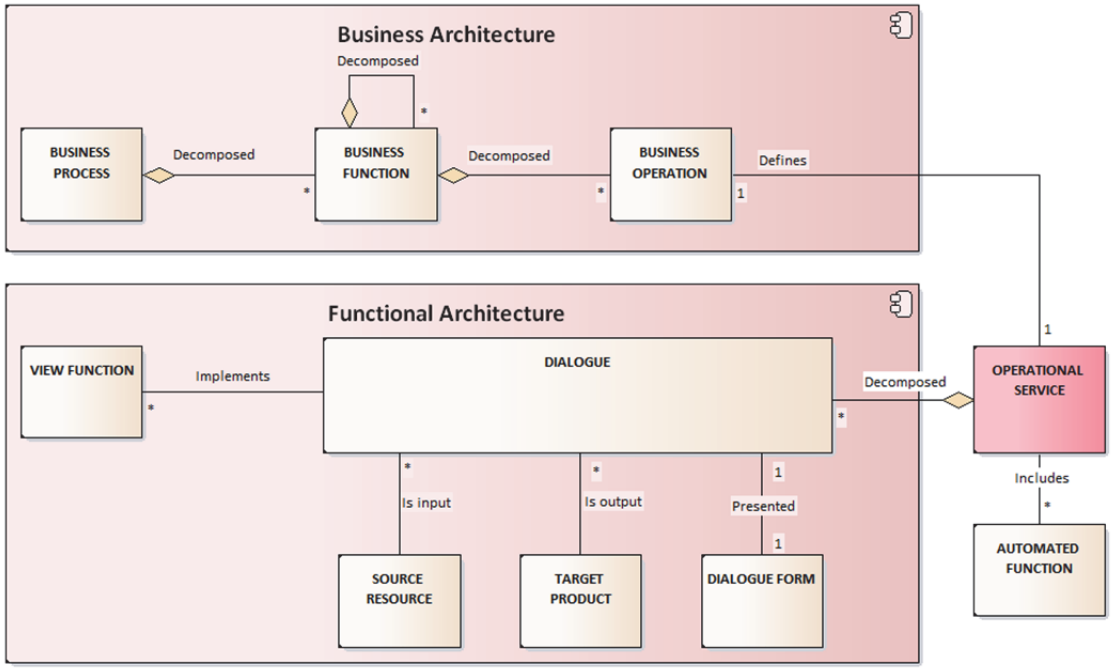

Figure 2 – Connection between business architecture and functional architecture

5 Component Architecture

The view functions resulting from the design of dialogs do not take into account the component architecture of the system. These functions concern only the presentation level of the IS. Representation of the IS architecture in the form of interacting functional components (subsystems and external systems) makes it possible to find out how the view functions implement, to determine the internal functions of the application logic and data management.

The component architecture establishes the composition and interaction of the functional components of the IS, determines the software modules and their distribution according to the functional components, details the functional requirements for the IS.

The main elements of the component architecture model are (Figure 3):

– functional components (subsystems and external systems), determined by the choice of an architectural template (or a composition of architectural templates) and the external environment of the system;
– software modules, which are the structural parts of the functional components of the IS and are determined by the decomposition of the view function.

The design of component architecture begins with the definition of the component structure of the IS and of which represent the system as the set functional components (subsystems and external systems). The modular composition of the functional components of the system is determined by the decomposition of the connecting elements of the functional

architecture - the view functions of IS, performed on the selected component structure. The decomposition of the view function may include previously defined modules for reuse, i.e. there is a many-to-many relationship between view functions and system modules.

In ACM, to determine the modular composition of IS, Sequence Diagrams are used, which are generated for the view functions of each dialogue described at the level of functional architecture. Functional components are used as lifelines in diagrams. Thus, design allows you to define a complete set of software modules for all functional components.

6 Definition of class methods

ACM offers a technology for developing a data architecture (see [13] and [14]), but an exposition of this technology is beyond the scope of this work. We proceed from the fact that the data architecture in one way or another is developed at the level of the ER model and includes all the necessary attributes, and the entities are distributed among the elements of the component model. The task is to determine the data architecture classes methods based on the generated modular structure: for each module of the component model, the order of its implementation within the framework of the object-oriented IS model is determined.

Thus, the connecting element that determines the seamless transition from component architecture to data architecture are the software modules of the component model. Class methods are determined by the decomposition of each module on the presented data architecture (Figure 3). As with the component model, there is a many-to-many relationship between modules and class methods, corresponding to the reuse of methods to implement modules. Class methods are also defined using Sequence Diagrams in which the lifelines correspond to the classes of the data model.

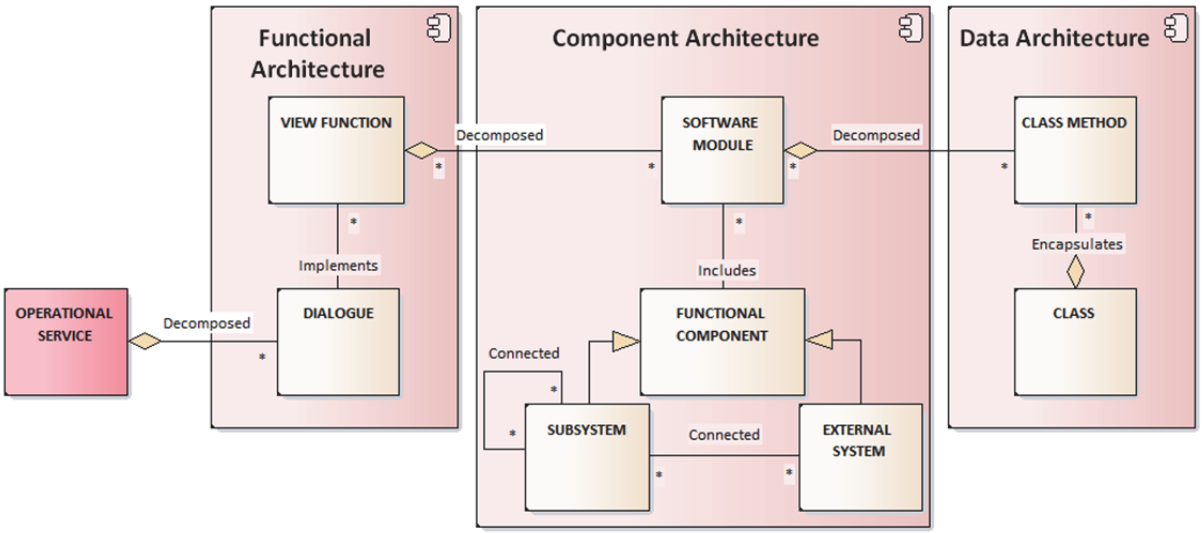

Figure 3 – Functional architecture, component architecture, data architecture and their relationship

7 Technological Architecture

Technological architecture is a model representing the technical infrastructure of IS, including solutions in the field of computing and telecommunications infrastructure.

Technological architecture, in the context of this work, represents the distribution of elements of component architecture (deployment) over various hardware and determines the necessary interfaces between objects of technical architecture. The choice of architectural solutions for technological architecture is limited by system requirements, quality attributes, and requirements for external interfaces.

In terms of level of abstraction, technological architecture is superior to component architecture. Figure 4 shows that the connecting element between these architectures is the element of the technological architecture corresponding to the hardware. Connection reflect the location of functional components on the IS hardware.

Architectural representations of technological architecture are typically performed using Deployment Diagrams. The description of the deployment diagram objects may include either requirements for the characteristics of the hardware-software tool, or an indication of a specific device model.

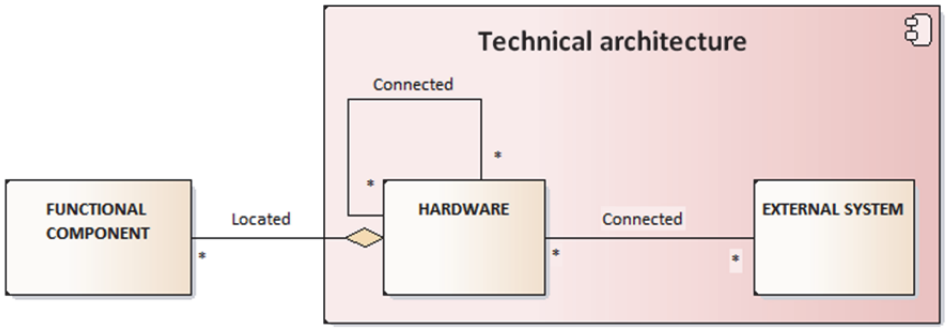

Figure 4 – The relationship of component architecture and technical architecture

8 Conclusion

The method considered in this work provides a seamless transition from business architecture through the decomposition of the operational services to the functional architecture of the IS, which defines the dialogs of the system and the view functions. The decomposition of the view functions defines the software modules of the component architecture and, finally, the modules are decomposed into classes methods of the data architecture. In other words, each architectural representation is derived from the architecture of the previous level of abstraction.

With this approach, the completeness of the functional implementation of the IS is ensured, since the decomposition of the business process allows you to accurately formulate the user requirements for the automated functions. Further design of functional components at various levels of the abstract description of IS is essentially a reasonable conclusion of the necessary and sufficient functionality of the IS, which avoids errors associated with technological gaps between architectural models, insufficient or excessive functionality.

The advantages shown by the ACM in real-world design and maintenance of IS include ensuring transparency of the compliance of the design result with the requirements set by the customer by simplifying the validation and verification processes. The presence of tracing between the elements of architectural models allows for the rapid localization of necessary changes and improvements for the release of new versions of IS. The regulation of the development process, the relationship of the architectural description with artifacts, taking into account the possibility of generating design and operational documents based on architectural models [11], provides a significant reduction in the design time and making necessary changes to IS.

In conclusion, I would like to outline topics that were not reflected in this work. Further research is required by the relationship between ACM and service-oriented architecture (SOA), namely the identification of services and the implementation of the component architecture of IC on the SOA template. In general, working with external information resources, whether it's web services or library classes, requires study in the context of the ACM.

Acknowledgments

The author expresses sincere gratitude to the professor, Doctor Boris Pozin, who provided invaluable assistance in the preparation of this article.

References

[1] R. Sessions. A Comparison of the Top Four Enterprise-Architecture Methodologies, 2007. [Online]. Available: http://rogersessions.com/library/white-papers#the-it-complexity-crisis-danger-and-opportunity.

[2] E. Shteingart and A. Burmistrov. Obzor i sravnitel'naya kharakteristika metodologiy razrabotki arkhitektury predpriyatiy (Review and comparative characteristics of methodologies for the development of enterprise architecture) [in Russian], St. Petersburg State Polytechnical University Journal. Economics, no. 3(245), pp. 111-129, 2016.

[3] The Open Group. The TOGAF Standard, version 9.2, 2019. [Online]. Available: https://www.opengroup.org/togaf.

[4] GERAM: Generalised Enterprise Reference Architecture and Methodology: Version 1.6.3, 1999. [Online]. Available: URL: http://www.cit.gu.edu.au/~bernus/taskforce/geram/versions/geram1-6-3/v1.6.3.html.

[5] FEA Consolidated Reference Model Document, Federal Enterprise Architecture Framework version 2, 2015. [Online]. Available: https://web.archive.org/web/20141030032759/http://www.whitehouse.gov/sites/default/files/omb/assets/egov_docs/fea_v2.pdf.

[6] The Open Group (1999–2006). ADM and the Zachman Framework, Online, TOGAF 8.1.1, 25 Jan 2009. [Online]. Available: http://pubs.opengroup.org/architecture/togaf8-doc/arch/chap39.html.

[7] Olaf Zimmermann, Pal Krogdahl and Clive Gee. Elements of Service-Oriented Analysis and Design, 2004. [Online]. Available: http://www.ibm.com/developerworks/webservices/library/ws-soad1/.

[8] N. M. Josuttis. SOA in Practice, Sebastopol, CA: O'Reilly, 2007.

[9] I. Graham. Object-Oriented Methods. Principles and Practice. Third Edition, Boston: Addison-Wesley, 2001.

[10] G. Tsiperman. Primeneniye metoda adaptivnoy klasterizatsii dlya proyektirovaniya slozhnykh informatsionnykh sistem (The use of adaptive clustering method for the design of complex information systems) [in Russian], in Proceedings of the 3rd International Scientific and Practical Conference Actual Problems of System and Software Engineering (APSPI — 2013, Moscow), Moscow, 2013.

[11] G. Tsiperman. Automation documenting the functionality of the information system using the method of adaptive clustering [in Russian], Highly available systems, vol. 11, no. 3, pp. 70-80, 2015.

[12] ISO/IEC/IEEE 42010:2011. Systems and software engineering — Architecture description.

[13] G. Tsiperman. Razvitiye metoda adaptivnoy klasterizatsii putem formirovaniya kontseptual'noy modeli obyekta informatizatsii (Evolution of the method of adaptive clustering by forming an informatization object conceptual model) [in Russian], in Proceedings of the 5th international scientific-practical conference Actual problems of system and software engineering (APSSE-2015), Aachen, 2017.

[14] G. Tsiperman. Training method for development of data model for information systems, in Proceedings of the XXII International Scientific Conference Enterprise Engineering and Knowledge Management, Moscow, 2019.

[15] ISO/IEC 12207:2008. Systems and software engineering — Software life cycle processes.