https://doi.org/10.31891/csit-2022-2-3
UDC 004.051

Andrii KOPP, Dmytro ORLOVSKYI, Dorukhan ERSOYLEYEN
National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine

AN APPROACH TO APPLICATIONS ARCHITECTURE MODELS ANALYSIS

A relevant problem of applications architecture model analysis was considered in this paper. Its significance is defined by the fact that designed blueprints of software systems should be thoroughly checked for all potential inefficiencies in order to avoid additional effort and costs for defect correction in later project stages. As a result, the research goal was defined as detecting strong and weak points in software design solutions via applications architecture model analysis. The research objective was set at the process of analyzing applications architecture models, and the research subject was set at the software solution for analyzing applications architecture models. Existing software tools for applications architecture modeling and analysis were defined based on an examination of general software development problems for applications architecture model analysis. The ArchiMate enterprise architecture modeling language was chosen as the standard representation of applications architecture models to be analyzed because there are nearly no alternatives to ArchiMate language for architectural description of enterprise application models that are standardized, supported by most diagramming software, and exchangeable. The domain of applications architecture models analysis was discovered, an approach to analyzing applications architecture models was proposed, a software solution for analyzing applications architecture models was designed and developed, and it was used to evaluate applications architecture models that represent web development patterns. The analysis results could be used by system or software architects to estimate the suitability of applications architecture solutions for ongoing projects, detect flaws in specific architectural patterns, and reduce effort and costs in later project stages.

Keywords: applications architecture, application model, model analysis, enterprise architecture, software solution.

Андрій КОПП, Дмитро ОРЛОВСЬКИЙ, Дорухан ЕРСОЙЛЕЄН
Національний технічний університет «Харківський політехнічний інститут», Харків, Україна

ПІДХІД ДО АНАЛІЗУ МОДЕЛЕЙ АРХІТЕКТУРИ ЗАСТОСУНКІВ

У цій роботі було розглянуто актуальну проблему аналізу моделей архітектури застосунків. Її значення визначається тим, що розроблені проєкти програмних систем повинні бути ретельно перевірені на наявність усіх потенційних недоліків, щоб уникнути додаткових зусиль і витрат на виправлення дефектів на наступних етапах проєкту. Отже, метою дослідження є визначення сильних і слабких сторін проєктів за допомогою аналізу моделей архітектури застосунків. Метою дослідження є процес аналізу моделей архітектури застосунків, а предметом дослідження – програмне рішення для аналізу моделей архітектури застосунків. Існуючі програмні засоби для моделювання та аналізу архітектури застосунків були визначені на основі огляду загальних проблем розробки програмного забезпечення для аналізу моделей архітектури застосунків. Мова моделювання архітектури підприємства ArchiMate була обрана як стандартне представлення моделей архітектури застосунків, які підлягають аналізу, оскільки майже немає альтернатив мові ArchiMate для архітектурного опису моделей корпоративних застосунків, які стандартизовані, підтримуються більшістю програмного забезпечення для створення діаграм, а також є придатними для обміну. Розглянуто предметну область аналізу моделей архітектури застосунків, запропоновано підхід до аналізу моделей архітектури застосунків, спроєктовано та розроблено програмне рішення для аналізу моделей архітектури застосунків, за допомогою якого були проаналізовані моделі архітектури застосунків, що представляють собою шаблони веб-розробки. Результати аналізу можуть бути використані архітекторами систем або програмного забезпечення для оцінки придатності рішень щодо архітектури застосунків для поточних проєктів, виявлення недоліків у конкретних архітектурних шаблонах та зменшення зусиль і витрат на наступних етапах проєкту.

Ключові слова: архітектура застосунку, модель застосунку, аналіз моделі, архітектура підприємства, програмне рішення.

Introduction

According to the IEEE standard (1471-2000), architecture is the fundamental organization of a system, embodied in its components, their connections with one another, and the environment, as well as the rules controlling its design and development [1]. Because the modern enterprise is a complex system comprised of several interconnected areas, the enterprise architecture describes the components, their relationships, and the principles underlying this system [2]. Enterprise architecture development aims to identify explicit links and dependencies between various organizational domains such as business architecture, information architecture, applications architecture, and technical infrastructure. Architectural development is concerned with describing the parts that comprise an enterprise and how they interact in order to improve the understanding and vision required to successfully design a business architecture and information technology. According to industry practitioners, the most significant but also most complex organizational area is the relationship between business architecture and application, and IT (Information Technology) architecture [1].

Leading organizations create application and IT architecture with industry-proven technologies that decrease architectural documentation duplication, shorten development cycle times, and provide consistent vocabulary that promotes consistency in architectural descriptions. Some of these technology providers have entered the Gartner Magic Quadrant for Enterprise Architecture Tools' Leadership [3].

One of the first tools to support the business architecture technique was ARIS Business Architect (Software AG). The ARIS product has its own methodology and does not support TOGAF, the Zachman Framework, or similar frameworks. The ARIS (Architecture of Integrated Information Systems) methodology is a comprehensive approach to developing and analyzing business process models, as well as modeling the overall enterprise architecture. The ARIS technique (also referred as "ARIS House") distinguishes five sorts of representations: organizational, functional, process, data description, and outputs. ARIS includes integration modules for the SAP system. Among the significant flaws is the ARIS closed metamodel, which prevents changes to the approach to business architectural management and the introduction of new types of objects [4]. ARIS prioritizes business process management and business architecture over application and IT architecture, which are viewed as just supporting the execution of business operations and are not at the heart of ARIS methodology and its software solutions.

The MEGA Suite (MEGA International) is a comprehensive enterprise architectural management solution that includes tools for modeling, control, transformation, communication, project planning, and strategic migration from as-is to as-should. Unlike ARIS, MEGA is a highly configurable meta-modeling tool that supports international TOGAF, DoDAF, and Zachman Framework standards, as well as modeling in ArchiMate, BPMN (Business Process Model and Notation), and UML (Unified Modeling Language) notations, and contains libraries of industry standards eTOM, ITIL, and APQC [4]. By our opinion, the one advantage of MEGA over ARIS is support for ArchiMate and UML modeling standards; nonetheless, this software solution is still focused on the management prospect of EA rather than the application and IT domains in particular.

Visual Paradigm, a significant participant in the software modeling area, has begun to support the ArchiMate modeling language [5]. The vendor of Visual Paradigm claims that their product is a powerful and intuitive diagramming tool for architecture modeling that supports: drag-and-drop editing interface; precise shape positioning with alignment guide; many formatting options for shapes and connectors; and color categorization of shapes [5]. Visual Paradigm has debuted its online ArchiMate modeling software, which is marketed as the most capable and user-friendly corporate architecture modeling software on the market [6]. It has an easy-to-use user interface and drag-and-drop capabilities, which reduces modelers' learning curve when creating ArchiMate diagrams. This online program also supports Microsoft Visio import and interaction with Microsoft Office. PDF and graphic formats are among the export possibilities. The most significant advantage of Visual Paradigm Online for ArchiMate modeling is that it does not require downloading or user registration and login – diagrams may be built on the go; all that is required is to access the product's web page. Obviously, such a product is far better suited for application and IT architecture design than ARIS, because it not only supports ArchiMate (while ARIS only supports its own notation for application and technical enterprise modeling), but also UML diagramming capabilities to explode ArchiMate application components into detailed low-level software descriptions.

In addition to the commercial products discussed, Archi, an open source cross-platform tool that supports The Open Group TOGAF and ArchiMate approaches, is worth considering. The Archi enterprise modeling tool is a modular framework built on top of the Eclipse integrated development environment (IDE), allowing architects or software developers to write plugins in Java [4]. Because of its free availability, this program can be used in place of the paid ARIS and MEGA solutions for working on minor, particularly educational, projects. Archi's open source distribution enables practically anyone to improve its features at no additional expense [7].

Despite the ease of use of Visual Paradigm Online for ArchiMate modeling, this tool has a significant drawback: it only allows the user to make and publish designed application and IT architecture solutions in PDF or graphical image formats, which are fine for human reading but cannot be processed by computers. In turn, ArchiMate supports the exchange file format enabling diagram export and interoperability with other modeling applications. This format is based on XML (eXtensible Markup Language), but it only follows a specific schema [8]. In terms of analytical capabilities, the examined products are more concerned with diagramming than with analyzing developed architectural solutions. Archi includes built-in validation for EA models, whereas this technique just checks for inconsistent modeling element usage (missing connections, duplicated or unused elements).

Problem statement

In general, applications architecture modeling is a phase of requirements analysis in the software development life cycle (SDLC). As a result, the following relevant activity inputs and outputs have been discovered and displayed on the context IDEF0/SADT (Structured Analysis and Design Technique) diagram shown below (Fig. 1).

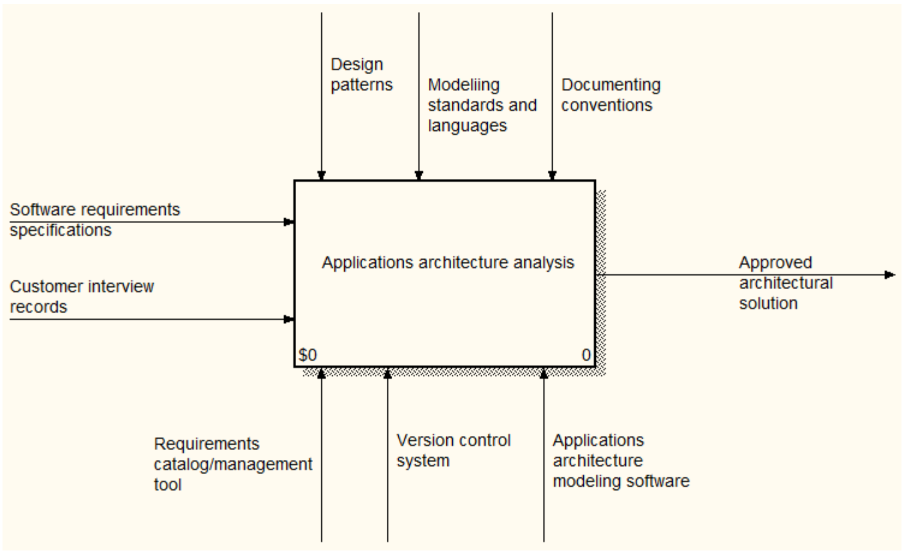

Design patterns
Modeliing standards and languages
Documenting conventions
Software requirements specifications
Customer interview records
Applications architecture analysis
Approved architectural solution
$0
0
Requirements catalog/management tool
Version control system
Applications architecture modeling software

Fig. 1. The context diagram of the applications architecture models analysis process

The context diagram shown above provides a basic overview of the entire applications architecture models analysis and how the software system that should be implemented can support this activity. Clearly, the proposed activity is a complex business process comprised of the following sub-activities: analyze requirement specifications, create blueprints for applications architecture, model and analyze applications architecture, brainstorm to identify architectural flaws and bottlenecks; provide a solution to the product owner.

Fig. 2 shows the decomposed IDEF0 diagram of the applications architecture models analysis process.

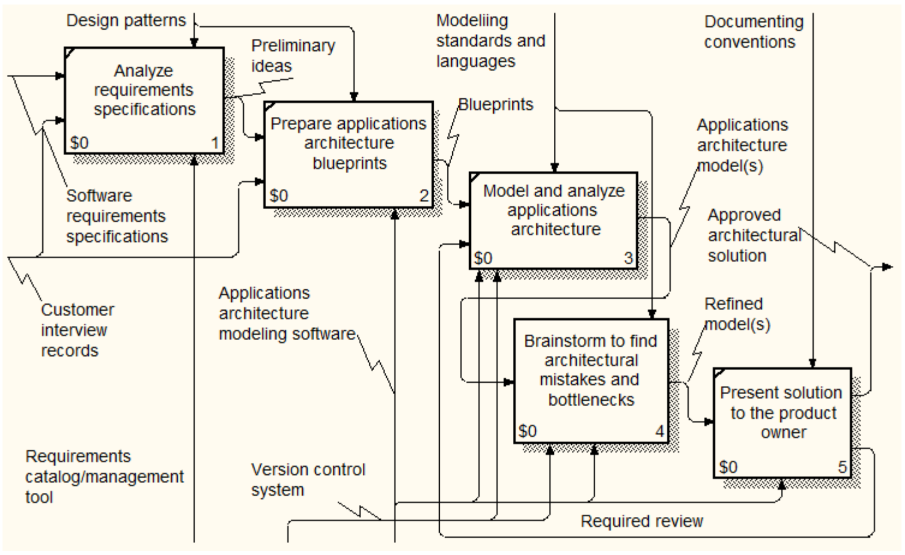

Design patterns
Preliminary ideas
Modeliing standards and languages
Documenting conventions
Analyze requirements specifications
$0
1
Prepare applications architecture blueprints
$0
2
Blueprints
Applications architecture model(s)
Model and analyze applications architecture
$0
3
Software requirements specifications
Approved architectural solution
Customer interview records
Applications architecture modeling software
Refined model(s)
Brainstorm to find architectural mistakes and bottlenecks
$0
4
Present solution to the product owner
$0
5
Requirements catalog/management tool
Version control system
Required review

Fig. 2. The decomposition diagram of the applications architecture models analysis process

As shown on the context diagram (see Fig. 1) and the IDEF0 decomposition diagram (see Fig. 2), the following inputs and outputs are used by and produced by the considered business process:

– the software requirements specifications and other customer interview records are used as inputs for the preliminary design of the applications architecture;

– design patterns, modeling standards and languages, as well as documenting conventions for applications architecture modeling are used to retrieve the architectural solution;

– all of the considered activities are carried out using: a requirements catalog or even management system (e.g. Jira Software for large agile projects), applications architecture modeling software (e.g. ArchiMate or Visual Paradigm), and the version control system (usually decentralized are used now, such as Git).

Using the business process model (Fig. 2), it was discovered that the sub-process associated with brainstorming architectural mistakes and bottlenecks is a bottleneck of the applications architecture models analysis process. Following the completion of such an activity, the refined architectural solutions are demonstrated to the product owner or another customer's representative. This means that step 4 (see Fig. 2) will need to be improved with the addition of a specialized software solution for analyzing applications architecture models.

Fig. 3 depicts the IDEF0 decomposition diagram of the considered activity (step 4) with the introduced software solution to be developed.

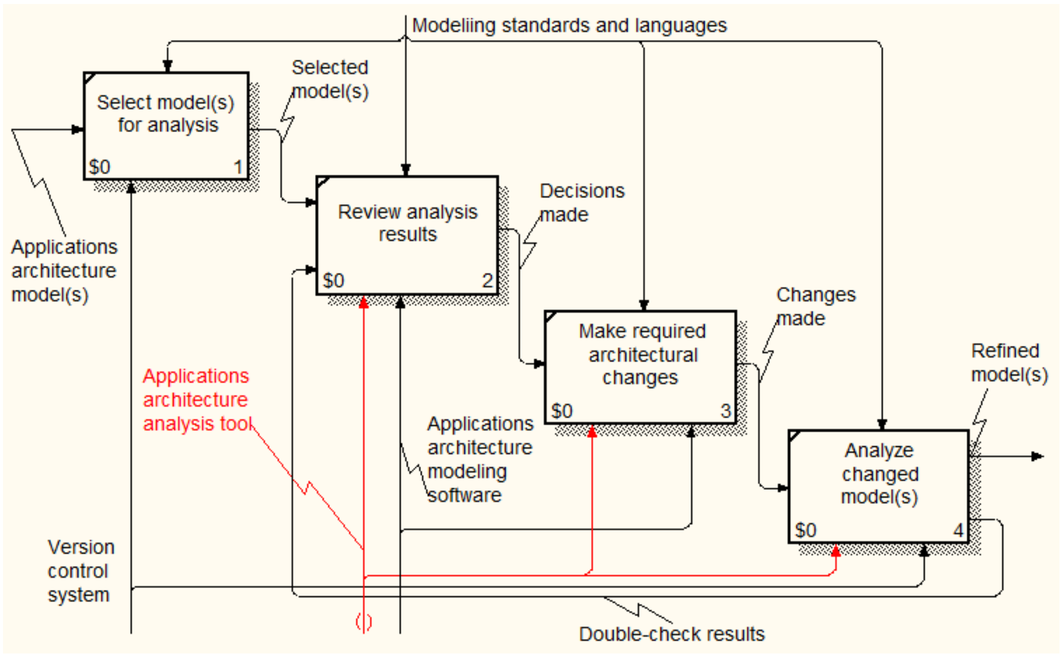

Modeliing standards and languages
Select model(s) for analysis
$0
1
Selected model(s)
Review analysis results
$0
2
Decisions made
Make required architectural changes
$0
3
Changes made
Analyze changed model(s)
$0
4
Refined model(s)
Applications architecture model(s)
Applications architecture analysis tool
Applications architecture modeling software
Version control system
Double-check results

Fig. 3. Decomposition diagram of the brainstorming sub-process to find architectural mistakes and bottlenecks

The sub-process, as depicted in the decomposition diagram (Fig. 3), includes the following tasks:

1) choose an applications architecture model (or models) for analysis;

2) review analysis results (after the analysis procedure is completed);

3) make necessary architectural changes (based on decisions made when analysis results were obtained and taken into consideration);

4) analyze changed applications architecture models (in order to check whether they require additional analysis and changes).

The special arrow in red indicates that the software solution for applications architecture models analysis will be used to obtain initial recommendations after analyzing the original model, as well as to perform double-check analysis of models modified based on analysis results.

Proposed approach to applications architecture models analysis

The ArchiMate applications architecture model represents a graph data structure that is widely used in computer science and its applications in the domain of software engineering [9]. For graph-based data structures, such as ArchiMate applications architecture models, link analysis (or network analysis) methods commonly used in system analysis can be used to evaluate system structure and draw conclusions about its properties and features.

In mathematics (graph theory), a graph is defined as a structure that represents a collection of objects, with some pairs of objects interconnected. Objects that correspond to some terms, concepts, or other mathematical abstractions are referred to as “vertices” (also known as nodes or points), and each connected pair of vertices is referred to as a “edge” (also sometimes called links, arcs, or lines). A graph is typically represented in schematic (visual or rather graphical) form as a set of points or circles for vertices connected by arcs (if the links are directed) or edges [10].

For example, a segment of an applications architecture model created with the ArchiMate language (see the upper part of Fig. 4) could be represented using the directed graph shown below (see the bottom part of Fig. 4).

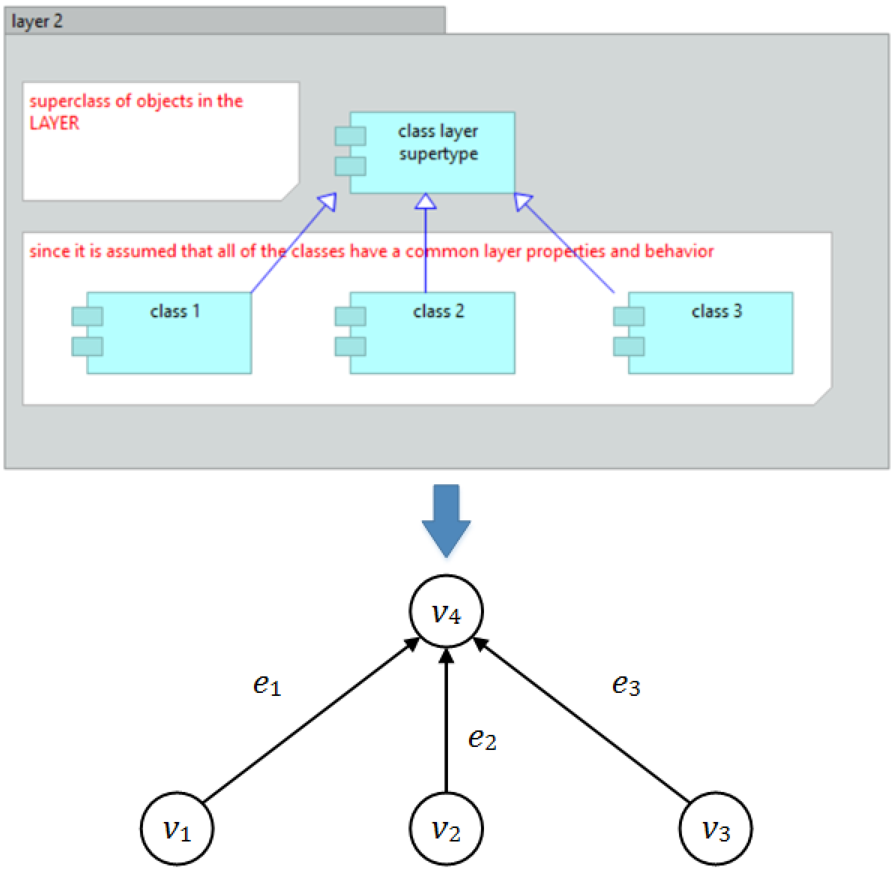

layer 2
superclass of objects in the LAYER
class layer supertype
since it is assumed that all of the classes have a common layer properties and behavior
class 1
class 2
class 3
v_4
e_1
e_2
e_3
v_1
v_2
v_3

Fig. 4. An example mapping between an applications architecture model and a graph-based data structure

The most basic and elementary measures of directed graphs are the following [10]:

– $|V|$ – is the number of vertices in a graph G;

– $|E|$ – is the number of edges in a graph G;

– $d_G(v)$ – is the degree of a vertex in a graph G, which represents the number of incident edges (vertex degree is sometimes referred to as “valency”);

– $d_G^{in}(v)$ – is the in-degree of a vertex in a graph G, which means the number of incoming edges;

– $d_G^{out}(v)$ – is the out-degree of a vertex in a graph G, which means the number of outgoing edges.

Using these elementary measures of a directed graph built on the basis of the ArchiMate applications architecture model, the following system analysis measures could be calculated:

1) connectivity:

$$A_G^{\Sigma} = \frac{1}{2}\cdot\sum_{v\in V} d_G(v); \tag{1}$$

2) resilience:

$$R_G = \frac{1}{2}\cdot\sum_{v\in V} d_G(v)\cdot\frac{1}{|V|-1} - 1; \tag{2}$$

3) centrality:

$$C_G = \frac{\sum_{v\in V}\left[\max_{v\in V}\{d_G(v)\} - d_G(v)\right]}{(|V|-1)\cdot(|V|-2)}; \tag{3}$$

4) irregularity:

$$\varepsilon_G = \sqrt{\sum_{v\in V}\left[d_G(v) - \frac{2\cdot|E|}{|V|}\right]^2}. \tag{4}$$

Using these measures (1 – 4), the following n-space vector $X = (x_1, x_2, \dots, x_n)$ [11] could be obtained as a “footprint” or “image” of a specific ArchiMate applications architecture model. In our case, the elements of vector X are determined by the previously discussed measures, hence, $n = 4$. Consequently, applications architecture is based on design patterns. Obviously, those patterns, like any other pattern, have advantages and disadvantages. Patterns are used implicitly when designing applications architecture. They are mostly derived from the knowledge and experience of system engineers. However, architectures that appear to be faultless on “blueprints” may contain hidden threats during software implementation, testing, and even maintenance.

As a result, in order to avoid extra costs and efforts in the late SDLC phases, it is necessary to recognize design patterns within created applications architecture models [12]. When compared to a specific applications architecture model represented also using the n-space vector $S = (s_1, s_2, \dots, s_n)$ [11], design patterns described as images X could be recognized using similarity measures. Since we need to check a given model for matching with multiple patterns, the set of design pattern images $P = \{X_1, X_2, \dots, X_m\}$, where m is the size of the design pattern collection used as "ideal" images, could be considered [11].

It is proposed to use the Euclidean distance to assess the similarity of design pattern images and applications architecture models under consideration [11]:

$$D(S, X_i) = \sqrt{\sum_{j=1}^{n} \left(x_{ij} - s_j\right)^2}. \quad (5)$$

Obviously, other similarity and distance measures could be used instead of (5), but for the current proof-of-concept implementation, stopping at the most well-known Euclidean distance measure is sufficient.

The pattern with the greatest similarity, i.e. the shortest Euclidean distance, should be used as the reference:

$$X^* = \arg\min_{X_i, i=\overline{1,m}} \{D(S, X_i)\}, \quad (6)$$

where X^* – is the pattern chosen to serve as a reference to provide suggestions.

Applications architecture design patterns discovered using (6) should be used to generate such recommendations based on the benefits and drawbacks of these patterns. Models of applications architecture describe software architecture as a system of interconnected application components [13].

Therefore, the following system design patterns [14], [13] can be considered:

1) if applications architecture model is the most similar to the sequential pattern, then following recommendation should be obtained "Development: easy, Cost: inexpensive, Flexible: yes, Reliability: moderate, Extension: easy, Robust: no";

2) if applications architecture model is the most similar to the ring pattern, then following recommendation should be obtained "Development: difficult, Cost: moderate, Flexible: no, Reliability: high, Extension: easy, Robust: no";

3) if applications architecture model is the most similar to the radial pattern, then following recommendation should be obtained "Development: easy, Cost: expensive, Flexible: yes, Reliability: high, Extension: easy, Robust: yes";

4) if applications architecture model is the most similar to the tree pattern, then following recommendation should be obtained "Development: easy, Cost: moderate, Flexible: yes, Reliability: high, Extension: easy, Robust: no";

5) if applications architecture model is the most similar to the mesh pattern, then following recommendation should be obtained "Development: difficult, Cost: expensive, Flexible: no, Reliability: moderate, Extension: difficult, Robust: yes".

All of the patterns under consideration should then be accompanied by their respective image vectors for use in applications architecture model analysis – $P = \{X_1, X_2, \dots, X_n\}$. Recognized design patterns should then be demonstrated with their obtained similarity measure values, each of which demonstrates the closeness or vice versa – incompatibility (for those patterns that were unexpectedly suggested as the analysis results) of the designed applications architecture model with the best practices. Such "suspicious" cases should then be displayed to a user (e.g., a system architect or analyst) to aid in the decision-making process for the architecture design of the software solution.

Design and development of the software solution

Let us now describe the proposed architectural solution's structure. We can begin with a description of IT infrastructure as the foundation for all proposed software systems. This layer includes the following structure elements:

– VCS (Version Control System) repository, which is a catalog in a file system that is under VCS control, and all changes in this catalog are tracked by VCS software, such as Git or Subversion;

– VCS file system, which is a built-over traditional file system that extends it with specific operations for version control management of stored files in a repository; (e.g. commit, push, pull, clone operations etc.).

The following behavior elements of the proposed architectural stack's IT infrastructure are:

– the process of software (and user) interaction with the VCS repository of applications architecture models;

– the services that VCS repository capabilities offer to software and users for accessing stored applications architecture ArchiMate models.

ArchiMate architecture models are stored as XML (eXtensible Markup Language) files in the OMG (Object Management Group) open exchange file format as information elements of the IT infrastructure of the depicted architectural solution.

The following structural architecture elements of the proposed architecture solution belong to the applications layer:

– Microsoft Power BI application (free desktop version) used to demonstrate analysis results as the user-interface solution for end users;

– software component (it is planned to create Java enterprise web application) that implements applications architecture models analysis approach considered before.

According to the proposed solution, behavioral software application elements are:

– the process of applications architecture models analysis;

– services of applications architecture models analysis provided to system analyst or system architect.

Java objects, also known as Java Beans, are information elements of the depicted architectural solution's applications layer. The proposed solution's business architecture layer depicts the activities of end users (system analysts or system architects), who should obtain recommendations for applications architecture model improvement via analytical reports to which Java objects are translated and displayed via Power BI. The developed solution's software architecture should be client-server in nature.

Spark Framework, a simple and expressive Java web framework built for rapid development, was used to implement web-based API (Application Programming Interface). Spark's goal is to provide an alternative for Java developers who want to create web applications that are as expressive as possible while using as little boilerplate as possible. Spark Framework is designed with a clear philosophy to not only make you more productive, but also to improve your code under the influence of Spark's sleek, declarative, and expressive syntax [15]. The Java application can pass JSON objects to the Power BI user interface using the Spark Framework.

Power BI is a Business Intelligence (BI) and Data Visualization tool that converts data from various sources into interactive dashboards and BI reports. Power BI suite includes a variety of software, connectors, and services, including Power BI desktop, SaaS-based Power BI service, and mobile Power BI apps for various platforms. Business users use this set of services to consume data and create BI reports [16]. Because of these features and benefits, Power BI was chosen to implement a user interface that could contain applications architecture models analysis reports that could be shown to software and system architects.

Fig. 5 depicts the software components UML diagram of the software solution for applications architecture models analysis.

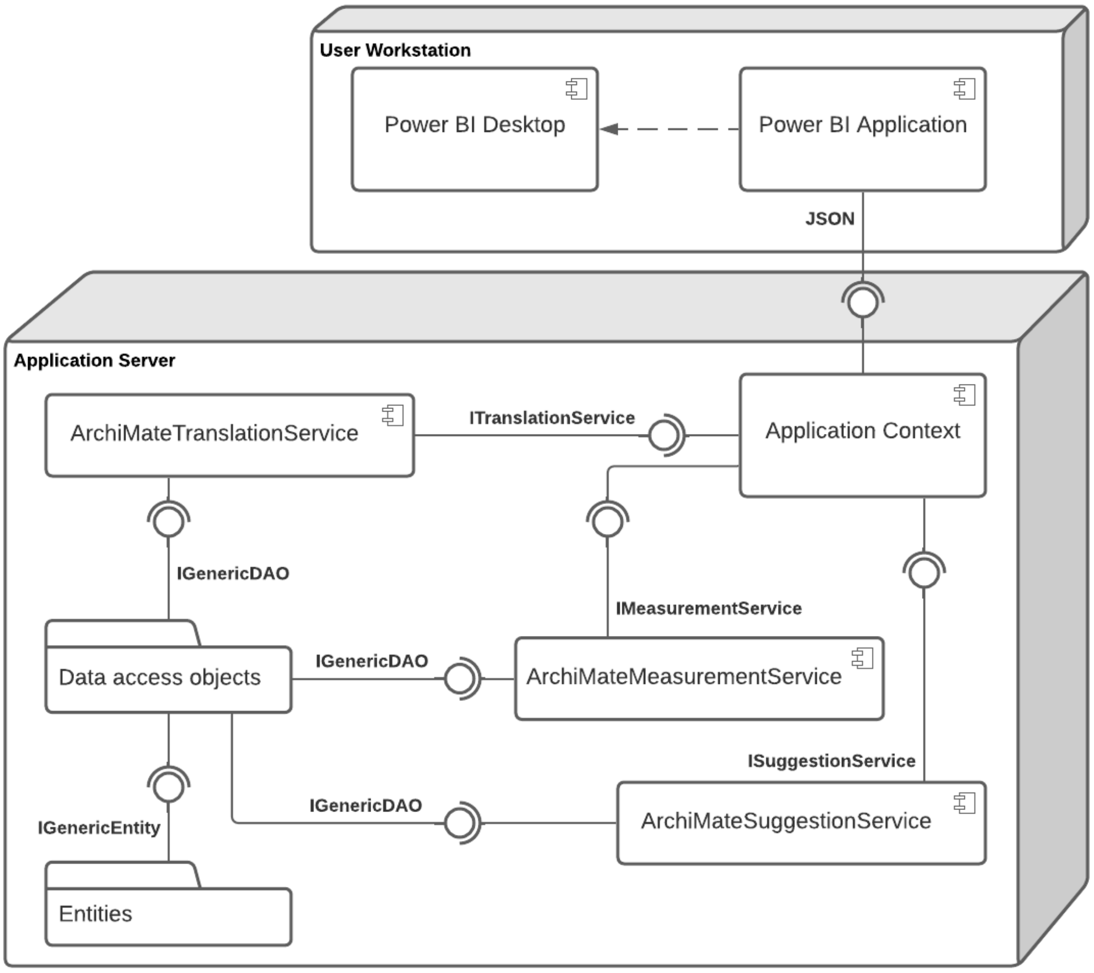


Fig. 5. The software solution's component diagram

The software solution is comprised of three major components, as shown in the diagram above (Fig. 5):

1) "ArchiMate Translation Service" component, which is in charge of extracting ArchiMate files from VCS repositories and translating them into Java objects for further processing;

2) "ArchiMate Measurement Service" component, which is in charge of calculating metrics (defined in section 2.1) for processed applications architecture models, the structure of which is already represented as Java objects;

3) "ArchiMate Suggestion Service" component, which is in charge of calculating the distance between applications architecture models and pre-defined structure patterns of systems analysis and producing appropriate recommendations.

As shown in Fig. 5, the considered software components rely on data access objects that are in charge of managing the data store and maintaining collections of Java objects that describe the structure of processed ArchiMate models. The user's workstation only contains the Power BI software and the respective analytical reporting application, which connects to the Java application via the JSON-based web API implemented with the Spark Framework.

Applications architecture models analysis using the software solution

To test the operability of the developed software solution, the following set of applications architecture models were chosen and presented in ArchiMate language, while originally belonging to the resource "Catalog of Patterns of Enterprise Applications architecture: Web Presentation Patters" [17]. Given the dominance of web applications today, it is important to detect the system design opportunities or threats of specific web development patterns.

Fig. 6 depicts the first part of the ArchiMate language's applications architecture models.

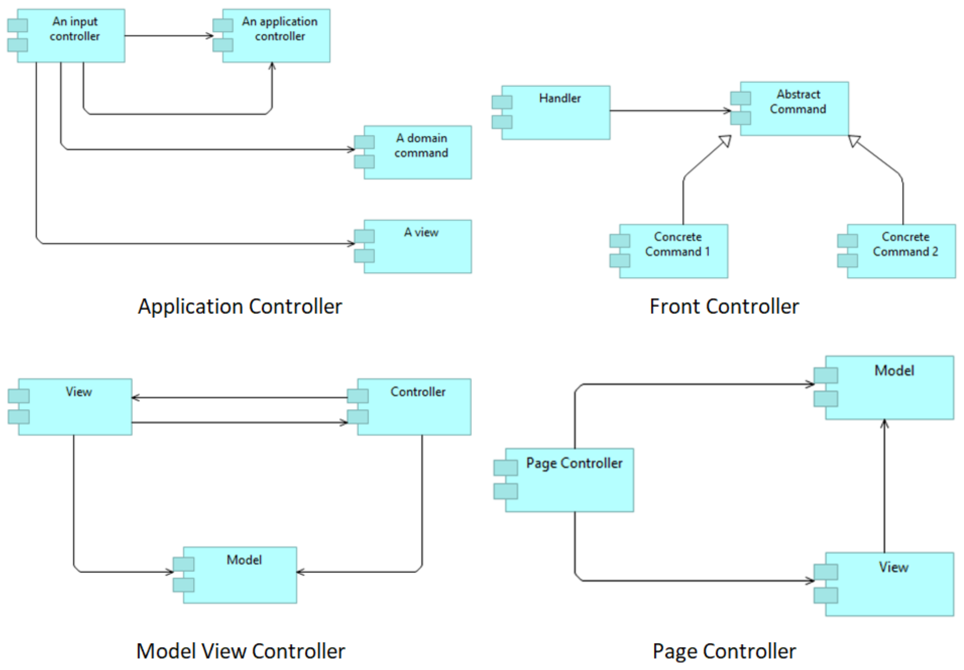


Fig. 6. The collection of applications architecture models that are used for analysis (first part)

Fig. 7 depicts the second part of the ArchiMate applications architecture models that have been prepared for analysis.

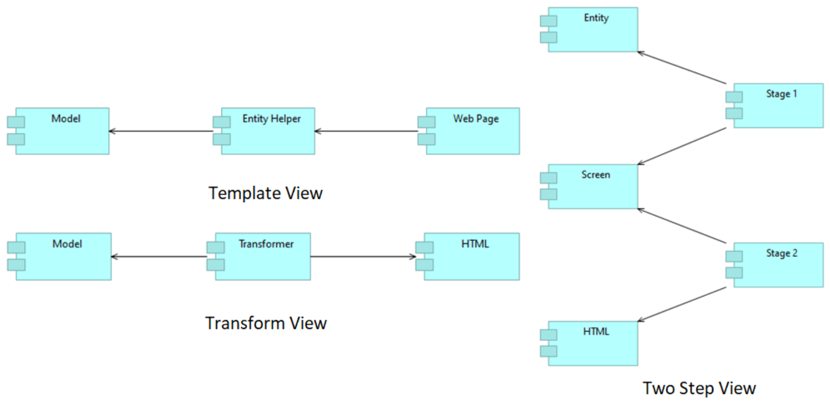


Fig. 7. The collection of applications architecture models that are used for analysis (second part)

Despite the fact that these models appear to be quite simple, they demonstrate essential web enterprise application development approaches, which analysis may be used by system or software architects to avoid defects and vulnerabilities in the future by making decisions during the design stage rather than fixing errors during the testing or even maintenance phases of the project's lifecycle.

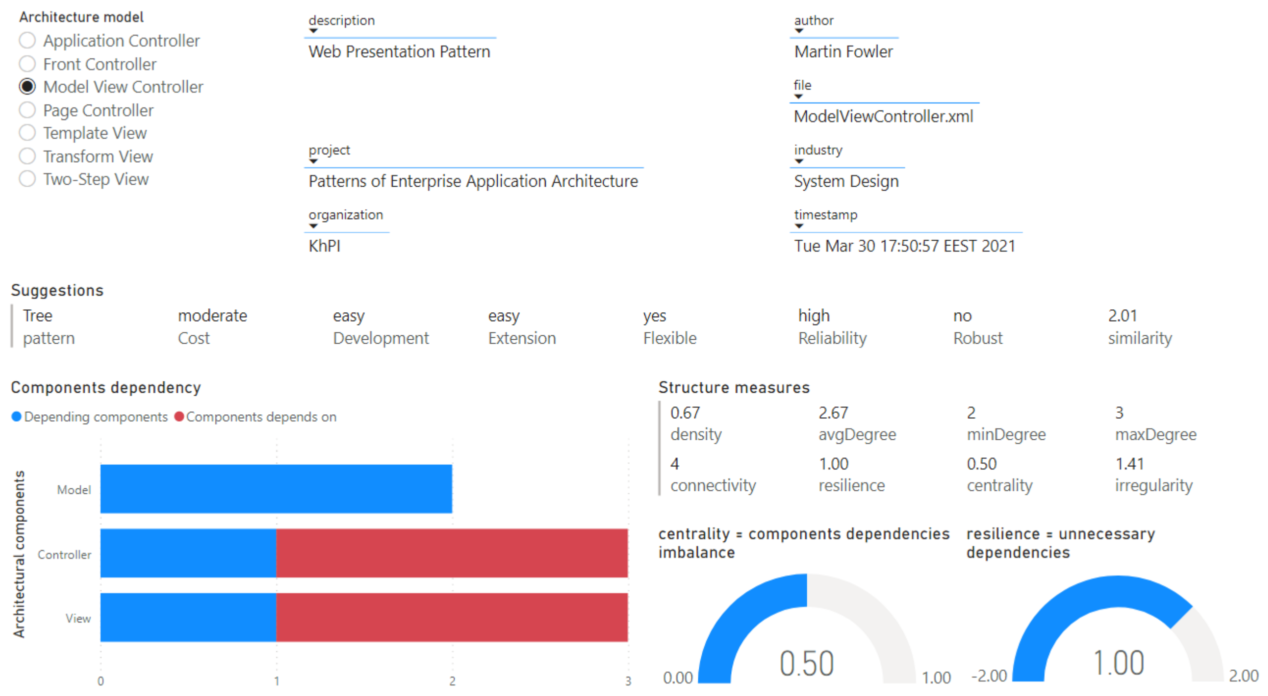
The recommendations on web enterprise application solutions to use should be provided. Let us start with the models that have flaws and are not recommended for use:

– "Application Controller" is costly to maintain, despite its robustness;

– development of a "Page Controller" solution is difficult, and it is not flexible;

– "Template View", "Transform View", and "Two-Step View" are inexpensive to maintain, but they provide moderate reliability.

As a result, "Model View Controller" (see Fig. 8) and "Front Controller" (which are sometimes considered as part of the "Model View Controller" solution) could be recommended as better solutions for enterprise applications architecture design, where high reliability and flexibility, moderate cost of maintenance, and ease of development and extension are important system design features despite a lack of robustness.



Architecture model
- Application Controller
- Front Controller
- [✓ Model View Controller]
- Page Controller
- Template View
- Transform View
- Two-Step View

description	Web Presentation Pattern
author	Martin Fowler
file	ModelViewController.xml
project	Patterns of Enterprise Application Architecture
industry	System Design
organization	KhPI
timestamp	Tue Mar 30 17:50:57 EEST 2021

Suggestions

Tree	moderate	easy	easy	yes	high	no	2.01
pattern	Cost	Development	Extension	Flexible	Reliability	Robust	similarity

Structure measures

0.67	2.67	2	3
density	avgDegree	minDegree	maxDegree
4	1.00	0.50	1.41
connectivity	resilience	centrality	irregularity

Components dependency
● Depending components ● Components depends on
Architectural components
Model
Controller
View
0
1
2
3
centrality = components dependencies imbalance
0.00
0.50
1.00
resilience = unnecessary dependencies
-2.00
1.00
2.00

Fig. 8. Obtained results for the "Model View Controller" model

The obtained results (see Table 1) are supported by nearly two decades of "Model View Controller" (MVC) dominance in enterprise application development as a result of its concept of never combining data and presentation.

Table 1

Detailed results for the "Model View Controller" model

Pattern	Cost	Development	Extensibility	Flexibility	Reliability	Robustness	Distance
Tree	Moderate	Easy	Easy	Yes	High	No	2.01
Connectivity		Resilience		Centrality		Irregularity	
4		1.00		0.5		1.41	

A plethora of web enterprise frameworks are either completely based on or can support MVC principles of application design and development.

Conclusions

In this paper, a relevant problem of applications architecture model analysis was considered. Its significance is defined by the fact that designed blueprints of software systems should be carefully checked for all potential inefficiencies in order to avoid extra effort and costs for defect correction in later project stages. As a result, the research goal was defined as detecting strong and weak points in software design solutions through the analysis of applications architecture models. The process of analyzing applications architecture models was designated as the research objective, and the software solution for analyzing applications architecture models was designated as the research subject.

Existing software tools that support applications architecture modeling and analysis were defined based on an analysis of general software development problems for applications architecture models analysis. The ArchiMate enterprise architecture modeling language was chosen as the standard representation of applications architecture models to be analyzed. The following tasks were completed in order to achieve the study's goal:

1) the problem domain of applications architecture models analysis was discovered;
2) the approach to analyzing applications architecture models was proposed;
3) the software solution for analyzing applications architecture models was designed and developed;
4) the software solution was used to examine applications architecture models that represent web development patterns.

The results of the analysis could be used by system or software architects to estimate the suitability of applications architecture solutions for ongoing projects, detect weak points in certain architectural patterns, and reduce effort and costs in later project stages.

In the future, this approach and software tool should be expanded to analyze not only applications architecture, but also remaining domains of enterprise architecture, such as business architecture (organizational structure, business processes, etc.) and technological architecture (IT infrastructure including system software, hardware etc.). In addition, the set of system design measures could be expanded in the future, and alternative distance measures could be used.

References

1. Minoli D. Enterprise Architecture A to Z: Frameworks, Business Process Modeling, SOA, and Infrastructure Technology / D. Minoli. – Boca Raton: Auerbach Publications, 2008. – 498 p.
2. Using ArchiMate and TOGAF to Understand the Enterprise Architecture and ITIL Relationship / M. Vicente et al. // Lecture Notes in Business Information Processing, 2013. – 148. – P. 134-145.
3. Products In Enterprise Architecture (EA) Tools Market // https://www.gartner.com/reviews/market/enterprise-architecture-tools, 10.03.2021.
4. Brocke J. Handbook on Business Process Management 2: Strategic Alignment, Governance, People and Culture / J. Brocke, M. Rosemann. – Springer, 2014. – 865 c.
5. ArchiMate Tool // https://www.visual-paradigm.com/features/archimate-tools/, 10.03.2021.
6. Online ArchiMate Software // https://ralpha-garcia.medium.com/online-archimate-software-93a29edaab4b, 12.03.2021.
7. Free ArchiMate Modeling Tool Archi // https://mikethearchitectblog.wordpress.com/2011/01/07/free-archimate-modeling-tool-archi/, 12.03.2021.
8. ArchiMate® Model Exchange File Format for the ArchiMate 3.1 Modeling Language // http://www.opengroup.org/xsd/archimate/, 20.03.2021.
9. Kopp A., Orlovskyi D., Ersoyleyen D. An approach to analysis of ArchiMate applications architecture models using the software coupling metric // Bulletin of National Technical University "KhPI". Series: System Analysis, Control and Information Technologies. – 2021. – No. 2 (6). – P. 67-72.
10. Trudeau R. Introduction to Graph Theory / R. Trudeau // Courier Corporation, 2013. – 224 p.
11. Sharma M. Analysis of Distance Measures in Content Based Image Retrieval / M. Sharma, A. Batra // Global Journal of Computer Science and Technology, 2014. – 14, No. 2. – P. 1-7.
12. Kopp A. M., Orlovskyi D. L., Ersoyleyen D. Applications architecture analysis based on design patterns and image recognition // MicroCAD-2021, 2021. – P. 14.
13. Kopp A., Orlovskyi D., Ersoyleyen D. General Issues of Applications Architecture Domain Design // Texas Journal of Multidisciplinary Studies, 2021. – No. 1 – P. 106-112.
14. Bisht N. Analytical study of different network topologies / N. Bisht, S. Singh // International Research Journal of Engineering and Technology, 2015. – 1, No. 2. – P. 88-90.
15. Spark // https://sparkjava.com/, 24.04.2021.
16. Power BI // https://www.tutorialspoint.com/power_bi/index.htm, 26.04.2021.
17. Catalog of Patterns of Enterprise Applications architecture // https://martinfowler.com/eaaCatalog/, 20.01.2021.