Journal of Computing and Security

Research Article, 2023, Volume 10, Number 2 (pp. 61–81)

http://www.jcomsec.org

Software Architecture Tools- A Classification and Survey with Recommendation for an Organization

Hassan Rashidi [a,*], Zahra Rashidi [b], Zeynab Rashidi [c]

[a] *Faculty of Statistics, Mathematics and Computer Sciences, Allameh Tabataba'i University, Tehran, Iran.*
[b] *School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran.*
[c] *Department of Instructional Technology, Allameh Tabataba'i University, Tehran, Iran.*

ARTICLE INFO.

Article history:
Received: 29 May 2023
Revised: 31 August 2023
Accepted: 09 September 2023
Published Online: 08 January 2024

Keywords:
Software Development, Software Tool, Software Architecture.

ABSTRACT

With the rise of cloud infrastructures, micro-services, frameworks, and reference architectures for every conceivable domain and quality attribute, someone might think that architectural knowledge is hardly needed anymore. But all the architect of today needs to select from the rich array of tools and infrastructure alternatives out there, instantiate, configure them, and create an architecture. Software architecture tools mean any software that helps automation and create architecture, according to requirements. The purpose of these tools is to reduce human effort, speed up software development, and increase reliability. This paper aims to perform a literature review of software architecture tools and to propose architectures for the requirements of the Organization of Small Industries and Industrial Towns of Iran (OSIITI). We surveyed more than 50 software architecture tools for use in practical situations and large-scale projects such as OSIITI's needs. The results of this survey identified five classes, namely (a) Modeling Tools to model architectures; (b) Code-Based Tools (Diagrams-As-Code) to perform syntactic and semantic consistency checking of the models; (c) Automated Tools to generate executable source code automatically that implements the models; (d) Diagramming Tools and (e) Icons-Based Tools to support for trace links between models and requirements or models and tests interfaces. For each class, several software tools are provided with their major features. These classes and tools are very helpful for organizations such as OSIITI that want to develop software, in both small and large-scale projects. A couple of architectures, based on layered and service-oriented patterns are proposed for OSIITI.

1 Introduction

In today's digital world, businesses need technology and knowledge of systems and their integrations to understand how the organization works. The expansion of the Internet and the use of software at a staggering speed has covered all fields on a small and large scale. With the rise of cloud infrastructures,

* Corresponding author.
Email address: hrashi@gmail.com(H. Rashidi)
doi:https://dx.doi.org/10.22108/JCS.2023.134862.1131
ISSN:2322-4460
https://dx.doi.org/10.22108/JCS.2023.137862.1131
ISSN: 2322-4460

micro-services, reference architectures, and frameworks, someone might suppose that architectural knowledge is scarcely needed anymore for applications and their quality attribute. But all of today's architecture must be selected and enhanced from a rich set of infrastructure tools and alternatives, instantiate, prototype, configure them, and create an architecture for specific applications. The simple principle of software architecture of a system is to construct a model to satisfy an organization's abstract/business goals, and that the architecture is a bridge between those abstract/business goals and the final result/a concrete system [1]. Software architectures play a fundamental role in the complex path from abstract goals to concrete systems, which can be designed, evaluated, and documented using well-known patterns along with techniques that will support the realization of these business goals. This complex path can be tractable and controlled by architectural tools.

This study began with the challenges facing the Organization of Small Industries and Industrial Towns of Iran (OSIITI) [2].This organization is responsible for supporting and developing small industries and studying and organizing the construction of industrial towns and industrial areas in Iran.

The most important missions of the organization in the current conditions are to provide the following services for its industrial units:

- **Mission 1**: Setting the stage for issuing technical and engineering services in the field of creating industrial investment infrastructures for industrial areas;
- **Mission 2**: Policy-making, planning, and drafting a strategic plan for the creation and development of technology towns, technology, and business clusters and business service centers as well as information technology complexes, and software services;
- **Mission 3**: Managing the provision of required financial resources and investment and trying to provide collateral for industrial units through investment guarantee funds;
- **Mission 4**: Facilitating, encouraging, and laying the foundation in order to create a suitable connection between its industrial units;
- **Mission 5**: Facilitating and encouraging the creation and development of R and D centers and innovation centers at the level of industrial fields;
- **Mission 6**: Policy-making and planning in the field of drafting service descriptions, guidelines, and regulations and draft by-laws for the creation and development of towns and industrial areas and their supervision;
- **Mission 7**: Planning for the empowerment of manpower and support of programs to provide specialized and skilled manpower needed by its industrial units;
- **Mission 8**: Policy-making, planning, and monitoring the creation of workshop complexes, service centers, and the creation of special economic zones under the responsibility.

According to the latest information from the deputy of OSIITI [2], the total number of stagnant units in the towns and industrial areas of Iran is 3000, and 151 industrial units. The main challenges of this organization are that: (a) around 44% of these units have the problem of lack of liquidity, (b) 16% lack of demand and market, (c) 10% wear out machines, (d) 10% weak technology, (e) 11% lack of raw materials, (f) 9% have legal problems and (g) 8% have infrastructure problems. They have several challenges in providing infrastructure for these problems, mainly in: (i) selling products and supplying services to domestic and foreign markets, (ii) obtaining raw materials, and (iii) providing capital for development. To respond to the challenges above, this organization must seek technologies and platforms that are used to collect, store, analyze, and access data as well as help organizational users to make better business decisions.

This paper is motivated to survey the software architecture tools and to propose general architectures for OSIITI. The remaining sections of this paper are structured as follows. In Section 2, we present the well-known patterns for software architecture and then we review the latest research devoted to the matter. In Section 3, we classify the major tools for software architectures. In this section, the major attributes of the latest tools are presented. In Section 4, we propose a couple of architectures for OSIITI. Section 5 is considered for the summary and conclusion.

2 Background and Related Works

The software architecture of an application is the set of structures required to reason about the application[1]. These structures encompass software elements, relations among them, and properties of both. There are architectural patterns that increase the productivity of architects with reuse. The architectural pattern captures the design structures of various systems and their elements. During the process of implementation and programming, developers come across similar problems several times inside a project, in their careers, and within the company. These reusable schemes address common software design challenges.

2.1 Well-Known Patterns

There are 14 software architecture patterns to be used in practice ([1], [3], [4]). The main pros and cons of

these patterns are described briefly below:

- **Client-Server Pattern**: This pattern is a peer-to-peer architecture for applications consisting of a client that requests a service and a server that provides the service. The general examples of this pattern include e-mail, file sharing, banking, and the World Wide Web. The main advantage of this pattern is that data and network peripherals are managed centrally, however, the server is costly.
- **Circuit-Breaker Pattern**: This pattern is used to minimize the effects of a threat in applications by redirecting traffic flow to another service. Although it helps make systems more resilient to prevent accidents, it requires the use of infrastructure management technology such as mesh service and sophisticated testing.
- **Command-Query-Responsibility-Segregation(CQRS) Pattern**: This pattern is suitable for applications where more database queries than data changes occur. It separates write and read activities for greater stability, performance, and scalability but requires more database technologies and thus may increase costs.
- **Controller-Responder Pattern**: This pattern is used to divide the architecture of applications into two components, (a) the controller manages the data and distributes the workloads, and (b) the responder replicates the data from the controller and produces results. The main advantage of this pattern is that it can read data from the responder without affecting the data in the controller, but if the controller fails, the system may lose data and need to restart.
- **Sourcing Pattern**: This pattern is suitable for systems that use real-time data. It sends a continuous stream of messages to a database, web server, report, or other target. It is highly flexible but requires a highly efficient and reliable network infrastructure to minimize latency.
- **Layered Pattern**: This pattern is suitable for desktop applications, e-commerce, and other systems that contain groups of subtasks that are executed in a specific order. The layered pattern makes it easy to make a system quickly, but the downside is that it's difficult to split the layers later.
- **Micro-Services Pattern**: This pattern is used to combine design patterns to create multiple services that work interdependently to create a larger system. Because each service is small, it's easier to update them as needed, but the complexity means that architects need more architectural expertise to make everything work properly.
- **Model-View-Controller (MVC) Pattern**: This pattern is used to divide a system into three components. The model comprises the data and the main functionality. The view shows data and interacts with the user. The controller manages the user's input and acts as an intermediary between the view and the model. This pattern enables the system to have different views, but its abstraction layers increase the complexity.
- **Publish-Subscribe Pattern**: This pattern publishes (sends) relevant messages to locations that subscribe to a topic. It is easy to make a configuration with this pattern but more challenging to test because the interactions between the publisher and subscriber are asynchronous.
- **Saga Pattern**: This pattern is used for multi-step transaction management, for instance, travel booking services. A "saga" consists of the various steps that must occur to complete a transaction. This pattern enables transactions (ideally with five steps or less) to be performed in loosely coupled, message-oriented environments but requires a lot of programming and can be complex.
- **Sharding Pattern**: This pattern is used to divide data in a database to speed up queries or commands. This ensures that storage space is consumed equally across all instances, but requires a skilled and experienced database administrator to effectively manage sharing.
- **Static-Content-Hosting Pattern**: This pattern optimizes web page load time in applications, in which static content (information that doesn't change often, such as an MP3 file or an author biography) is separate from dynamic content (such as stock prices). This pattern is very efficient for serving content and media that doesn't change often, but its downsides include data stability and higher storage costs.
- **Strangler Pattern**: This pattern is suitable when architects are willing to make incremental changes to a system. It puts the legacy system behind an intermediary to support incremental transformation, which reduces risk compared to making heavy changes. However, architects should pay considerable attention to routing and network management and ensure that architectures have a rollback strategy in place should problems arise.
- **Throttling (Speed-Control) Pattern**: This pattern is suitable for controlling how the speed of data flows to a target in applications. It is frequently used to prevent failure during a distributed denial of service attack or manage cloud infrastructure costs. To successfully use this pattern, architects need reliable redundancy mechanisms, and it is often used together with the circuit-breaker pattern to control service perfor-

mance.

2.2 Related Works

The development of international software with global standards requires many supports of software tools with special capabilities There is limited research devoted to classification and survey architecture tools. In this section, the latest research on these tools is reviewed.

Portillo Rodríguez et al. (2010) explained some desirable capabilities to support these tools in the field of internationalization and also how these capabilities are related to the main challenges in the international areas[5]. The authors surveyed the capabilities of these tools. The tools included in the survey were classified using ISO/IEC 12207 standard processes to determine which software development processes are supported by each tool. The classification of tools according to their capabilities is also included, where the specific capabilities of each tool are shown.

Majidi et al. (2010) did a survey and made a classification of software architecture[3]. This research studied the existing architectural patterns and surveyed the classification of the patterns. In this study, the advantages and disadvantages of three kinds of classifications are analyzed, and then several criteria for choosing a suitable pattern for different systems are proposed.

In recent years, mobile applications (apps) attracted many investments and interests. Apps are software applications or computer programs designed to execute on mobile devices such as tablets phones, or watches. Tavakoli et al. (2018) surveyed the tools used for the development and intelligent mining procedures behind mobile applications [6]. To gain a vision into the maturity of provision mining tools, this research also identified what challenges they are facing and found out what techniques are included in the tools for app development. The results of this survey provide useful knowledge for the development of intelligent apps with more effective mining techniques and tools.

Fregnan et al.(2019) surveyed software coupling relations and development tools [7]. The first goal of this study was to present a classification of different types of couple relationships in software components, along with criteria for their measurement. The second one includes providing an overview of the tools that have been proposed so far by the academic society of software engineering to derive these criteria. This study conducted a systematic literature review in software engineering. Publicly available scientific research databases were used to retrieve cited publications. These resources were queried using the keywords inherent to the software coupling. They considered publications from the years 2002 to 2017 and heavily cited earlier publications. A snowball method was utilized to retrieve the most relevant content. The results of this research showed there are four groups of coupling relationships, including semantic, structural, logical, and dynamic in software components. Moreover, it showed there is a fifth group of coupling relationships included in approaches as an independent group and criteria that have been developed for particular environments. Regarding the criteria in tools, this research identified three trends, emerging in recent years: scalability, extensibility, and the use of visualization methods. The research also found tools that extract criteria relevant to each coupling group. This research presents some directions for software coupling and some applications of coupling components to follow. Developing criteria for particular environments and code smell detection are presented for possible future research directions.

Ozkaya and Erata (2020) surveyed challenges faced by software specialists in their software modeling activities[8]. In this survey, responses from over 80 software specialists from 18 countries who developed software for different industries are collected. They focused on eight categories of challenges in software modeling: (a) management of the software complexity, (b) extending software modeling, (c) domain-specific modeling environments for software development, (d) developing formal models for software development, (e) analyzing software models, (f) separating of concerns in codes, (g) transforming software models, and (h) managing software models. The results showed separation of concerns is a minor challenge in the categories for specialists while analyzing models is the major challenge in the categories. Several concrete challenges in different categories have been detected, including (i) working the software modeling along with a steep learning curve, (ii) inconsistencies and updating the software tools when the software semantics is extended, (iii) evolving the domain-specific language tools with new requirements, (iv) creating the formal models for software semantics and translations of formal languages, (v) festering software models into separate perspective and analyzing the consistencies between different perspective in software models, (vi) transforming and synchronizing the software model in consistent ways, (vii) using model inspectors for formal and semi-formal analysis, and (viii) versioning software models.

Tian (2021) surveyed source code and software architecture [6]., which are two tangled artifacts that embody the dependent design decisions made at two levels of abstraction, low-level and high-level, for software development. The authors utilized a combined method of an online survey with 87 respondents from 37 countries and an interview with eight experts. The

results revealed that software specialists mainly discuss five features of relationships between software architecture and source code. Moreover, it shows a few software specialists have agreed features of relationships between software architecture and source code. Moreover, it shows a few software specialists have agreed on dedicated approaches and software tools for finding and analyzing the relations between software architecture and source code, regardless of identifying the importance of such information for enhancing a system's quality attributes, particularly reliability, and maintainability. The opinions obtained from this research showed that efforts and costs are the major obstacles that prevent software specialists from identifying, analyzing, and using the relationships between software architecture and source code. Moreover, the results have empirically recognized five features of relationships between software architecture and source code stated from the viewpoint of software specialists. The research suggested that an organized framework to manage the features of relations needs to be developed with dedicated approaches and tools.

Galster and Weyns (2023) did empirical research in software architecture with perceptions of the community[9]. They studied perceptions of the research community on (i) how empirical research is applied, (i) human participants, (iii) internal and external validity, and (iv) replications. They collected responses from 105 key players in architecture research via a survey and analyzed data quantitatively and qualitatively. The results showed although respondents do generally not prefer either quantitative or qualitative research, around 40% express a preference for various reasons. Moreover, it showed that professionals are the preferred participants so there is no consensus on the value of student participants. Also, there is no consensus on when to focus on internal or external validity. Most respondents value replications but acknowledge difficulties. A comparison with published research shows differences between how the community thinks research should be done. This research provided some evidence that consensus about empirical research is limited.

BehbahaniNejad and Rashidi (2023) proposed a novel architecture based on a business intelligence approach to exploit big data [10].In this paper, an architecture was proposed to integrate both Business Intelligence and Big Data architectures. To evaluate the proposed architecture, the research investigated business intelligence architecture and Big Data architecture. Then, it developed a Unified Modeling Language diagram for the proposed architecture. In addition, using the Colored Petri-Net, the proposed architecture is evaluated in a case study. The results show that the architectural system has a higher efficiency in performing all steps, average time, and maximum time compared to business intelligence architecture. The proposed architecture can help organizations and companies gain more value from their data sources as well as provide better support for managers in their decision-making.

Table 1 makes a summary of the related works on software architecture in recent years. In this table, the main features, advantages, and disadvantages of the previous research are shown.

3 Software Architecture Tools- A Classification

In this paper, we surveyed more than 50 software architecture tools. The results of this survey identified five classes, as shown in Table 2. In practical situations and large-scale projects in industry, we must model architectures (by Modeling tools), syntactic and semantic consistency checking of the models (via Code-Based tool), automatic generate executable source code that implements the models (by automated tools), support for trace links between models and requirements or models and tests interfaces (by diagramming and Icon-based tools). So, the tools can be classified as (a) Modeling Tools; (b) Code-Based Tools (Diagrams-As-Code); (c) Automated Tools; (d) Diagramming Tools; and (e) Icons-Based Tools[**?**]. For each class, the latest major tools found are in the right column of Table 2.

3.1 Modeling Tools

The first class of architectural tools is modeling tools that provide a fundamental capability of objects from which are reused and synced across several views, sometimes at different levels of abstraction with highly prejudiced validation. They are great for understand component dependencies and long-lived documentation with simple structured levels of detail. We did a survey of well-known tools in this field that can improve the productivity of affairs to a great extent in the modeling tools. The results of this survey are summarized in Table 3.

Table 1: A Summary of Related Works.

Author (Year) [Ref]	Main features	Advantages	Disadvantages
Majidi et al. (2010) [3]	It studied the existing architectural patterns and surveyed the classification of the patterns	It provided several criteria for choosing a suitable pattern for the different proposed systems	Lack of criteria for particular environments
Tavakoli et al. (2018) [6]	It studied the tools used for the development and intelligent mining procedures behind mobile applications	It provided useful knowledge for the development of intelligent apps	Study only the tools used for mobile Apps.
Fregnan et al. (2019) [7]	It surveyed software coupling relations and development tools	It showed there are four groups of coupling relationships, including semantic, structural, logical, and dynamic in software components	Lack of criteria for particular environments
Ozkaya and Erata (2020) [8]	It surveyed challenges faced by software specialists in modeling activities from 18 countries in different industries.	They focused on eight categories of challenges in software modeling	Dedicated to only the modeling activities.
Tian (2021) [11]	It surveyed source code and software architecture	It utilized a combined method of an online survey with 87 respondents from 37 countries and an interview with eight experts	Dedicated to only the relationships between software architecture and source code
BehbahaniNejad and Rashidi (2023) [10]	It proposed a novel architecture based on a business intelligence approach to exploit big data	The proposed architecture is evaluated in a case study. It used average time, and maximum time for the evaluation of business intelligence architecture.	Dedicated to only the Modeling Tools.
Galster and Weyns (2023) [9]	It studied perceptions of the research community on (i) how empirical research is applied, (ii) human participants, (iii) internal and external validity, and (iv) replications.	There is a comparison that shows differences between how the community thinks research	Without any classification of software tools.

Table 2: Classification of Tools for Software Architecture.

Category	Major Tools
ModelingTools [1]	*IcePanel, Enterprise Architect, Archi, Structurizr, Carbide, StarUML, Aplas, GenMyModel, Gaphor, Hackolade, VisualParadigm, Archipeg, Astah, MooD, Modelio*
Code-Based Tools (Diagrams-As-Code) [2]	*PlantUML, Structurizr, Ilograph, Graphviz, Mermaid, Diagrams, Diagram.codes, Web Sequence Diagrams*
Automated Tools [3]	Brainboard, Hyperglance, Hava, Archium
Diagramming Tools [4]	*Visio, LucidChart, Draw.io, Cloudcraft, isoflow, Archium, Terrastruct, Cloudockit, yuml, Cacoo, Cloudviz, Creately, Omnigraffle, Excalidraw, CloudSkew, Figma, yEd Live, Whimsical, Gliffy, Miro, Sketchboard, Mural, Sketch*

[1] https://softwarearchitecture.tools/#modelling-tools
[2] https://softwarearchitecture.tools/#code-based-tools
[3] https://softwarearchitecture.tools/#automated-tools
[4] https://softwarearchitecture.tools/#diagramming-tools

Table 2

Category	Major Tools
Icons-Based Tools [5]	*Google Cloud product icons, Azure icons, AWS icons, Kubernetes icons*

Table 3: Modeling Tools.

Tools [Ref.]	A Short Description	Main Features
Ice Panel [6]	A Collaborative C4 Modeling Tool, Designed to Increase Team Trust	• It supports to model of reusable relationships between objects. • It helps to sync software architecture updates across all diagrams. • It shows diagram overlays with showing multiple perspectives. • It supports interactive messages flowing through systems. • It can visualize the evolution of design decisions. • It speeds up onboarding and knowledge transfer without handholding.
Enterprise Architect [7]	An Enterprise Diagramming And Modeling Tool	• It provides a platform to support architects to stay in control of their workspace, manage their colleagues and team, enhance collaboration, and build confidence within their most complex projects. • It supports a unified view of complex systems, devising several viewpoints and several possible sub-systems. • It supports common models that can be accessed directly and securely by remote members of the development team through Enterprise Architect's Pro Cloud Server.
AArchi [8]	An Open Source Modeling Toolkit for Making Archimate Models and Sketches	• It contains an Archi modeling toolkit that is targeted toward all levels of Enterprise Architects for Modelers. • It is suitable to users: (a) those who look for an open source, a cross-platform modeling tool for applications, (b) with less experience in the modeling language, and (c) involve with the language in a TOGAF (The Open Group Architecture Framework) or other Enterprise Architecture framework
Structurizr [9]	A C4 Diagram and Documentation Tool	• It is specially designed and implemented to support the C4 model for visualizing software architecture. • It supports interactive diagrams with zoom-in/out and animation, it is also embeddable. • It can generate a diagram key/legend automatically for each diagram.
Carbide [10]	A Visual C4 Modeling and Diagramming Tool	• It is used for the trusted C4 model to design software systems and building architecture. • It supports all the features that architects want to have a great architecture.
StarUML [11]	An UML-Focused Modeling Tool	• It includes several useful Add-Ins with different functionalities. • It can generate source codes in programming languages and convert the codes into models, import Rational Rose files, and exchange modeling information with other tools using XMI.

[5] https://softwarearchitecture.tools/#diagram-and-cloud-icons
[6] https://icepanel.io/
[7] https://sparxsystems.com/
[8] https://www.archimatetool.com/
[9] https://structurizr.com/
[10] https://carbide.dev/
[11] https://staruml.io/

Table 3: Modeling Tools (continued)

Tools [Ref.]	A Short Description	Main Features
Aplas [12]	An Unparalleled Visibility Tool for Software Landscape	• It is an enterprise architecture software that supports businesses' software asset metadata for large-scale software project management and helps to develop source codes. • It supports the development team to produce, populate, and customize application, integration, along with creating indexes of fields, such as links, images, strings, numbers, and rankings.
GenMyModel [13]	A Collaborative Modeling Tool	• It helps to design software architecture and make business processes, quickly. • It supports to model of BPMN, UML, RDS, and creating Flowchart. • It enables users to design in the browser and visualize code.
Gaphor [14].	An Open Source Modeling Tool	• • It supports UML, SysML, RAAML, and C4 modeling applications. • It is fully compliant with the UML 2 data model (much more than a drawing tool for pictures). • It helps architects to visualize different aspects of a system quickly and create very complex models.
Hackolade [15]	A Visual Schema Design Tool for Data Storage and Data Exchanges	• It is a Visual Schema Design Tool for Data Storage and Data exchange. • It helps to make business sense of data with "Metadata-as-Code" and visual schema design for data storage and data exchanges.
VisualParadigm [16]	An Online Collaborative Modeling Tool	• It is a diagram maker, that supports collaboration, visual communication, and design tools anywhere and anytime. • It supports diagrams, charts, strategic analysis, and customer journey maps.
Archipeg [17]	A Combined Enterprise & Solution Architecture Tool	• It helps to connect technology with business, easily. • It helps cross-functional, technology-driven organizations connect the two worlds and make informed decisions.
Astah [18]	A Modeling Tool for UML, ER, Data Flow, Flowcharts Etc.	• It allows architects to visualize their ideas and software designs. • It builds diagrams quickly and effortlessly to provide a clear understanding among members of developers. • It supports drawing Flowcharts and building Data Flow Diagrams, UML, ER diagrams, mind maps, and more.
MooD [19].	A Modeling Tool with Flexible and Embedded Frameworks	• It can create an architecture of systems with a single, coherent working model, for optimal planning, flexibility, and agile management. • It provides a common language and a single reference so developers can do planning, Modeling, testing, and operating critical and complex systems. • It helps to create modeling scenarios for optimal operations.

[12] https://aplas.com/
[13] https://www.genmymodel.com/
[14] https://gaphor.org/
[15] https://hackolade.com/
[16] https://online.visual-paradigm.com/diagrams
[17] https://archipeg.com/
[18] https://astah.net/
[19] https://www.moodsoftware.co.uk/

Table 3: Modeling Tools (continued)

Tools [Ref.]	A Short Description	Main Features
Modelio [20]	An Open Source Modeling Tool	• It is an open-source modeling tool for UML2 and BPMN2. • It provides a broad-focused range of standards-based functionalities for system architects, business architects, analysts, designers, and software developers.

3.2 Code-Based Tools (Diagrams-as-code)

The second class of architectural tools is code-based tools that are used to design diagrams by combining a programming language and text that can be put in the storage of source control, allowing for integrations with workflows and development practices. We did a survey of well-known tools in this field that can improve the productivity of affairs to a great extent in the diagrams and cades. The results of this survey are summarized in Table 4.

Table 4: Code-Based Tools.

Tools [Ref.]	A Short Description	Main Features
PlantUML [21]	A Text-Based UML Diagramming Language with Simple Syntax	• It is a component that allows architects to quickly write The following non-UML diagrams are also supported
Structurizr [22]	A Text and Code-Based C4 Diagram and Documentation Tool	• It supports the C4 model for visualizing software architecture. • It supports interactive, animatable, and embeddable diagrams with zoom-in/out showing. • It creates a key/legend for diagrams automatically.
Ilograph [23]	An Interactive Diagramming Tool with YAML	• It supports interactive diagrams that are a revolutionary new way to document and explain systems with amazing clarity and detail. • It supports showing all the relevant details of systems without the constraints of a traditional diagram. Systems are not two-dimensional. • With this tool, systems must be viewed from multiple perspectives to be fully understood. Ilograph diagrams make this possible natively. Change is inevitable. • It supports diagrams that are defined exclusively using YAML, so they're easy to create and even easier to maintain.
Graphviz [24]	An Open Source Text Graph Visualization Tool	• It is an open-source graph visualization tool, that represents structural information as diagrams of abstract networks and graphs. • It can create visual interfaces for different areas and domains, creating software engineering, database and web design, machine learning bioinformatics, and networking.
Mermaid [25]	A Text and Code-Based Diagram and Visualization Tool	• It supports JavaScript-based diagramming. • It helps to create and modify diagrams dynamically, charting and rendering Markdown-inspired text definitions.

[20] https://modelio.org/
[21] https://plantuml.com/
[22] https://structurizr.com/
[23] https://www.ilograph.com/
[24] https://graphviz.org/
[25] https://mermaid-js.github.io/mermaid

Table 4: Code-Based Tools (continued)

Tools [Ref.]	A Short Description	Main Features
Diagrams [26]	A Draw Cloud System Architecture in Python Code	• It lets architects to draw the cloud system architecture in Python code. • It helps to describe or visualize the existing system architecture as well. • It supports diagrams as Code that allows architects to track the architecture diagram changes in any version control system.
Diagram.codes [27]	A Text-Based Diagram and Visualization Tool	• It used simple text to quickly create and share diagrams with the developer team. • It improves documentation, planning, and everyday communication with powerful tools for automatic diagram generation. • It allows architects to choose from the growing diagram catalog.
Web Sequence Diagrams [28]	A Text-Based Sequence Diagramming Tool	• It is a web-based application that allows the user to use a proprietary language to define sequence diagrams. • It helps to create a form of interaction diagram that displays objects as lifelines on the form, with their interactions over time as messages drawn as arrows from the source lifeline to the target lifeline.

3.3 Automated Tools

The third class of architectural tools is automation tools that are used to help automate the management of live infrastructure or the production of diagrams from live infrastructure. We did a survey of well-known tools in this field that can improve the productivity of affairs to a great extent in automation. The results of this survey are summarized in Table 5.

3.4 Diagramming Tools

The fourth class of architectural tools which are diagramming tools that are used to make freeform diagrams with no validation, allowing visual explanation of the developer's ideas. They are great for quick and easy short-term diagrams. We did a survey of well-known tools in this field that can improve the productivity of affairs to a great extent in the diagramming. The results of this survey are summarized in Table 6.

[26] https://diagrams.mingrammer.com/
[27] https://diagram.codes/
[28] https://websequencediagrams.com/

Table 5: Automated Tools.

Tools [Ref.]	A Short Description	Main Features
Brainboard [29]	A Collaboratively Design Tool That Generates and Deploys Terraform Code	• It is a solution for enterprise cloud infrastructure and to building an end-to-end cloud management solution. • It supports developers in building cloud architectures, with best practices and security by design.
Hyperglance [30]	An Automatic Aggregate Cloud Inventory and Interactive Diagrams Tool	• It is built by cloud professionals, for cloud professionals. • It helps to save money, reduce risk, automate tasks, and stay compliant thanks to round-the-clock cloud optimization mixed with powerful & interactive inventory visualizations.
Hava [31]	An Automated Interactive Cloud Diagrams Tool from Cloud Vendors	• It provides a set of logically arranged infrastructure diagrams grouped by VPCs or resource zones, showing all details of the resources and their connectivity. • It provides several options to show the attributes for each resource, such as security groups, connections, subnets, ingress/egress IPs, • It helps architects identifying anomalies, review cost forecasts and exporting diagrams for audit, managing, and compliance purposes.
Archium [32]	An Interactive Modeling Tool with Auto-Creation From Tracing Data.	• It can generate architecture models with interactive diagrams on the go. • It has a data flow visualizer that traces the flow of data in the system. • It enables micro-services to organize APIs and functions through an automated service catalog. • The distributed tracers of Archium enable storage of high cardinality telemetry for analysis of performance, post getting requests from individuals. • It does not ask for access to any production system, it just asks permission to extract data from the API's distributed tracer. Then the collected data is inspected by Archium Agent and incorporated into Archium SaaS.

Table 6: Diagramming Tools.

Tools [Ref.]	A Short Description	Main Features
Visio [33]	A Collaborative Diagramming Tool with Vast Templates	• It is a visual collaboration tool for development teams. • It helps architects to create professional diagrams anytime, anywhere. • Visio in Microsoft 365 is available to Microsoft 365 commercial subscribers.
LucidChart [34]	A Flexible Collaborative Diagramming Tool for Multiple Purposes	• It is just innovating faster. • It is great for team collaboration because architects can have multiple people working on the same document, and it's always up to date. • It is used to build a lot of integrations that let architects bring in data quicker and build diagrams around the data.

[29] https://brainboard.co/
[30] https://hyperglance.com/
[31] https://hava.io/
[32] https://archium.io/
[33] https://microsoft.com/en-ca/microsoft-365/visio
[34] https://lucidchart.com/pages/solutions/engineering

Table 6: Diagramming Tools (continued)

Tools [Ref.]	A Short Description	Main Features
Draw.io [35]	A Flexible Free Diagramming Tool for Multiple Purpose	• It is a free online tool for creating process diagrams, flowcharts, org charts, ERD, UML, and network diagrams. • It works with Google Drive and Google Workplace • It works with OneDrive, Sharepoint Office 365 and, Microsoft Teams apps.
Cloudcraft [36]	An AWS Visual Designer Tool with Smart Components	• It helps to create a professional architecture diagram with the Cloudcraft visual designer and optimization for AWS with smart components. • It helps to import the existing AWS environment into a new project. • It helps to enhance a design iteratively and easily.
isoflow [37]	A Diagramming Tool for Communicating Network Architecture	• It helps to draw clear and concise network diagrams easily, without a designer.
Archium [38]	An Interactive modeling Tool with Auto-Creation from Tracing Data	• It is an interactive architecture Modeling software that can be used to create a model by tracing the distributed data from the system automatically. • It enables developers to make design decisions faster and better. Additionally, it identifies risks that are hidden in the system and can be a threat to system reliability.
Terrastruct [39]	A Tool For Diagram Abstractions and Iteration with Integrated Data	• It is a general-purpose tool for designing diagrams and coding like coding on Google Docs. • It helps to design complex systems, through a feature suite tailored to software engineers.
Cloudockit [40]	A Tool Can Automatically Create Diagrams and Docs on the Cloud	• It offers 2D/3D diagrams with multiple views, technical documentation and reports, advanced scheduling options, complete cloud monitoring, and compliance rules. • It is a tool in the Architecture Design Tools category of a tech stack.
yuml [41]	A Tool to Create and Share Simple UML Diagrams in Wikis, Forums, and Issue Trackers.	• It allows architects to create diagrams quickly by simply typing plain text. • It can easily create diagrams from an architect's CI scripts or code. • It simply requests a diagram/an image programmatically using a GET or a PUT. • It is used by tool vendors to integrate yUML with wikis, blogs, and script

[35] https://draw.io/
[36] https://cloudcraft.co/
[37] https://isoflow.io/
[38] https://archium.io/
[39] https://terrastruct.com/
[40] https://cloudockit.com/
[41] https://yuml.me/

Table 6: Diagramming Tools (continued)

Tools [Ref.]	A Short Description	Main Features
Cacoo [42]	A Collaborative Diagramming Tool With In-App Video and Chat	• It is a simple tool to use without sacrificing the features architects need • It helps to brainstorm with anyone to love together on the same diagram in real-time, from anywhere. • It is multi-user editing that lets architects see who's editing what exactly as they do it. • It supports dynamic charts easily to add important data. • It lets to create, edit, comment on, and chat about diagrams whether architects are in the same room or half a world apart.
Cloudviz [43]	A Tool Can Automatically Generate AWS Architecture Diagrams and Docs	• It is a cloud architecture visualization solutions provider. The product generates AWS architecture diagrams and documentation in a few clicks. • The features of the product include secure cloud connection, diagram generation, and documentation, automation and scheduling, synced data visualization, etc.
Creately [44]	A Collaborative Diagramming Tool with Vast Templates	• It helps architects to build powerful intuitive apps to solve several kinds of workflow and challenges in developing architecture. • It supports Custom Databases, visual workspaces, wiki and notes, Task Management, Multi-Player Collaboration, and Modeling and Diagramming.
[45]	A Collaborative Diagramming Tool with Vast Templates	• It is used to create graphics and visuals. • It has several features for application features to design along with a drag-and-drop WYSIWYG interface • It helps to annotate and create specifications for prototypes and mockups with documentation.
Omnigraffle [46]	A Diagramming, Prototyping and Design Tool	• It is powerful visual communication. • It is used for rapid prototyping and designing. • It helps to organize diagrams and communicate them visually. • It provides a precise way and quickly to create beautiful wireframes to explore ideas accurately.
Excalidraw [47]	An Online Collaborative Whiteboard Tool with Sketched Design	• It is a virtual whiteboard for sketching hand-drawn-like diagrams, running in the browser. • The cool thing about this tool is that it requires no setup; architects can start drawing immediately at the speed of thought.

[42] https://cacoo.com/
[43] https://cloudviz.io/
[44] https://creately.com/
[45] https://omnigroup.com/omnigraffle
[46] https://omnigroup.com/omnigraffle
[47] https://excalidraw.com/

Table 6: Diagramming Tools (continued)

Tools [Ref.]	A Short Description	Main Features
CloudSkew [48]	An Online Cloud Diagramming Tool	• It is an online diagram and flowchart editor without having to install any software. • It is used to draw Icons for GCP, AWS, CNCF, Azure, Kubernetes, Alibaba Cloud, Oracle Cloud (OCI), IBM Cloud, and more, which are already available in the app. • Architects don't have to search for and download the symbols separately. • It is upgraded to produce templates professional.
Figma [49]	A Collaborative Design, Prototyping, and Whiteboard Tool	• It is a web-based design tool with the capability to create mockups, animations, interaction, and professional prototypes. • It is used as a design system component library. • It is trusted for control versions of architectures with collaboration features, presentation mode, and code generation.
yEd Live [50]	An Online Diagramming Tool	• It is the browser version of the powerful yEd desktop application. • It is used without an installer and creates high-quality diagrams quickly and effectively. • It is built upon the diagramming library yFiles for HTML so that the automatic layout algorithms arrange large data sets.
Whimsical [51]	A Tool For Collaborative Workspace	• It is a collaboration suite tool designed to create flowcharts, sticky notes, documents, wireframes, mind maps, and more.
Gliffy [52]	A Collaborative Diagramming Tool With Atlassian Integration	• It is used to build documentation with its Atlassian Apps. • It was one of the original apps on the Atlassian Marketplace. • It is used for the deepest integrations with Atlassian's tools today. • It is an interactive diagram maker within Confluence. • It can toggle through layers and information in the diagram viewer.
Miro [53]	An Online Collaborative Whiteboard Tool	• It is a visual platform to connect, collaborate, and create an architecture of applications together. • It can be used by the development team to work in the office, remotely, or a mix of the two.
Sketchboard [54]	An Online Collaborative Whiteboard Tool	• It is a virtual whiteboard tool, designed to share ideas, collect feedback and visually collaborate on a digital workspace. • It supports project planning, idea management, and visual workflow management and creates diagrams, templates, and presentations.

[48] https://www.cloudskew.com/
[49] https://figma.com/
[50] https://yworks.com/products/yed-live
[51] https://whimsical.com/
[52] https://gliffy.com
[53] https://miro.com
[54] https://sketchboard.io

Table 6: Diagramming Tools (continued)

Tools [Ref.]	A Short Description	Main Features
Mural [55]	An Online Collaborative Whiteboard Tool	• It is a visual collaboration tool with easy use. • It helps to work with the architect's team in a shared and dynamic visual environment. • It helps to run productive meetings and workshops with Facilitation Superpowers. • It supports multiple ways to connect and communicate with the architect team.
Sketch [56]	A Collaborative Design And Prototyping Tool	• It is a Mac app for designers to create, team up, prototype, and more. • It helps to browse, give feedback, inspect, and handoff- in any browser.

3.5 Icons-Based Tools

The fifth class of architectural tools is Icons-Based tools which are used in architecture diagrams. These diagrams are a great way to communicate over design, deployment, and topology of a system. Moreover, icons, in general, and cloud icons, in particular, can help communicate design decisions and the relationships between components of a given workload. In this section, we surveyed well-known solutions in this field that can improve the productivity of affairs to a great extent in the software architecture. The results of this survey are summarized in Table 7.

4 Proposed Architectures for OSIITI

In this section, a couple of architectures for OSIITI, are proposed. These architectures are discussed with experts and proposed based on two different views/purposes. The first view is that OSIITI is a large organization and has a hierarchical chart with several layers. This architecture is based on layered patterns for technologies and platforms that are used to collect, store, analyze, and access data as well as help organizational users make better business decisions (see the Introduction of the paper). The second architecture is based on service-oriented to provide services for its industrial units and doing missions (see the Introduction of the paper). This view is that OSIITI must provide several services for too many numbers of stagnant units in the towns and industrial areas. The details of these architectures along with the challenges are described below.

[55] https://mural.co
[56] https://sketch.com/

Table 7: Icons-Based Tools.

Tools [Ref.]	A Short Description	Main Features
AWS icons [57]	An Official Icon Pack	• It covers the collection of Amazon Web Services (AWS) and Architecture Icons (formerly Simple Icons) that contain AWS product icons, resources, and other tools to help architects to produce diagrams. • It has several icon resources that are used by customers and partners to create architecture diagrams. • The icons are designed to be simple so that architects can easily incorporate them in their diagrams and put them in presentations, whitepapers, posters, datasheets, or any technical material.
Google Cloud product icons [58]	An Official Icon Pack	• It allows the representation of the actors, use cases, functions, and devices graphically involved in a Google Cloud solution created for an organization. • It covers several types of icons: GCP Big Data Icons, GCP Compute Icons, GCP Developer Tools Icons, GCP Extras Icons, GCP Identity and Security Icons, GCP Machine Learning Icons, GCP Management Tools Icons, GCP Networking Icons, GCP Storage and Databases Icons
Azure icons [59]	An Official Icon Pack	• It helps architects to design and make new solutions, as cores to the Azure Architecture Center's mission. • It supports the collection of Azure architecture icons containing Azure product icons. • It helps architects to build a custom architecture diagram for their next solution.
Kubernetes icons [60]	A Community Icon Pack	• It is used to standardize Kubernetes architecture diagrams for presentation. • It can create uniform architecture diagrams that improve understandability. • Each icon can be found in different formats PNG and SVG.

4.1 Proposed Architecture Based on Layered Pattern

Figure1 shows a proposed layered architecture where each layer in the system consists of several logically related components. The main reason to propose this pattern is that it is commonly used to represent how a system can be broken down into components, and each component provides substantial functionality to the system, like the structure of OSIITI. Moreover, this pattern is suitable for desktop applications, e-commerce, and other systems that contain groups of subtasks that are executed in a specific order, according to applications of OSTIITI.

Major web-based systems and mobile apps work based on event triggering. An event in the user interface, for example, a mouse click, initiates actions to execute the user's selection. Furthermore, in the layered system, the flow of control is from top to bottom. User events at higher layers initiate actions at that layer, which in turn trigger events at lower layers. In contrast, much information flows in the bottom-up direction in the system. Information is created at lower layers, then transformed at intermediate layers, and finally supplied to users at higher levels.

Table 8 describes the functionalities of each layer in the proposed architecture. The databases include several databases required by OSIITI, such as selling products and supplying services to domestic (DataBase1) and foreign markets (DataBase2), obtaining raw materials (DataBase3), and providing capital for development (DataBase4). These databases are specified in the introduction.

[57] https://aws.amazon.com/architecture/icons
[58] https://cloud.google.com/icons
[59] https://docs.microsoft.com/en-us/azure/architecture/icons
[60] https://github.com/kubernetes/community/tree/master/icons

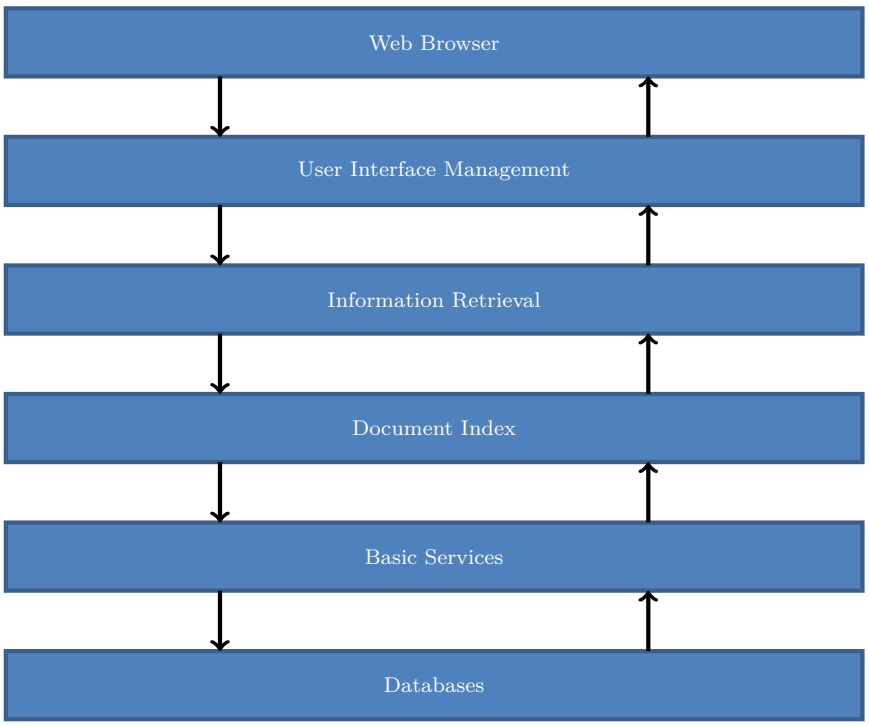

Figure 1. A Proposed Architecture Based on Layered Pattern for OSIITI.

Table 8: Functionalities of the Proposed Architecture, Shown in Figure2.

Layer	Major Functions
Web Browser	Inputs Entry, Local Input Validation, User Interaction, Local Printing
User Interface Management	User Authentication and Authorization, Web Page Generation, Form and Query Management
Information Retrieval	Retrieval of Documents, Right Management, Payment and Accounting, Search
Document Index	Creation of Index files, Index Management, Index Querying
Basic Services	Performing Database Query, Query Validation, User Account Management and Logging
Databases	OSIITI databases including DataBase1, DataBase2, DataBase3, DataBase4

4.2 Proposed Architecture Based on Service-Oriented Pattern

Figure2 shows the service-oriented architecture for the requirement of OSIITI. The main reason to propose this pattern is that OSIITI must provide several services (see the OSIITI's missions in the introduction) for its clients, including 3000 and 151 industrial units. Services in this architecture are components without a state, meaning they can be replicated and migrated from one computer to another. Moreover, many servers can provide several services. As the figure shows, there are six serves to perform OSIITI's missions (see the introduction). We propose 's1' to perform Mission 1, 's2' to perform Mission 2 and Mission 3, 's3' to accomplish Mission 4, 's4' to achieve Mission 5, 's5' to achieve Mission 6 and 's6' to perform Mission 7 and Mission 8. Usually, the service-oriented architecture is more scalable as demand increases and is more resilient to failure.

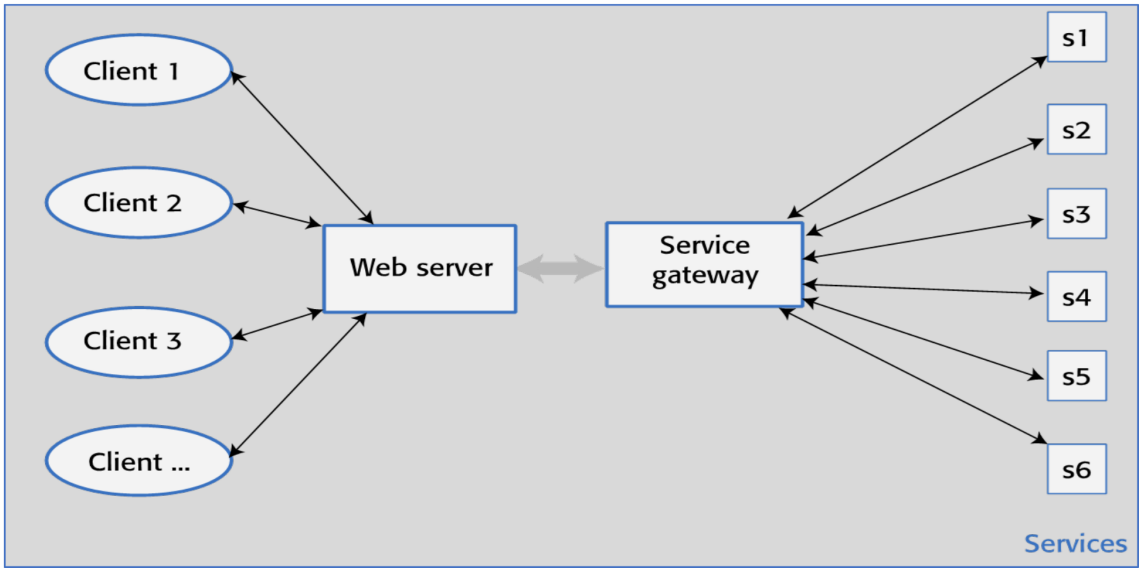

Figure 2. A Proposed Architecture for OSIITI, based on Service-Oriented Architecture.

In Figure2, the service gateway is a single access point and functions as a proxy for multiple services. This gateway supports routing, transformations, and ordinary processing across all the services. Ihe service gateway is a single module that manages the service requests from multiple clients (industrial units of OSIITI) and the services provided by the servers inside OSIIT. Four steps are usual for any service gateway:

- **Ordinary Processing**: When a message is received by the gateway, ordinary processing follows for all messages, such as creating and adding protocol-level headers or logging the message.
- **Service Identifying**: Any message that is processed by the gateway, must be recognized and its service type must be specified. For instance, the message is queried to decide whether it is a request for the service provider A, B, or C.
- **Endpoint Routing**: When a message is determined to be sent to a particular service provider, it is mapped to an addressable endpoint on the network so that the message can be sent to the service provider.
- **Specific-Service Processing**: Any processing required for the specific target service is performed.

There are three challenges in this architectural choice. These challenges are related to data, the frequency of changing components, and execution platforms. Some useful guidelines are provided in the following, to respond to these challenges.

- **Data Types and Data Updates**: If OSIITI wants to use structured data that can be updated by different system capabilities, it is usually better to have a single shared database that provides transaction management and locking. If data is distributed among services, OSIITI needs an approach to control consistency. This approach adds overhead to the system.
- **Frequency of Change**: If OSIITI anticipates that system components will be modified or replaced regularly, then separating these components as separate services will simplify those modifications.
- **System Execution Platform**: If OSIITI plans to deploy its systems in the cloud while accessing its users via the Internet, a service-oriented architecture is usually better to implement because this architecture supports scalability easily. If OSIITI wants to develop some specific products as business systems that run on local servers, a multi-tier architecture may be more applicable.

Table 9 describes the technology, design decisions, and comments on each decision for this architecture.

Table 9: Tool Options, Design Decisions, and Comments on OSIITI's Decisions.

Tool Options	Key Question	Comments and Recommendations
Open-source Tools	Are there any suitable open-source components that OSIITI could incorporate into its products?	• If OSIITI uses open-source software, its advantages are that OSIITI can reuse rather than develop new software tools. It will reduce development time and costs. • The disadvantage of using open-source tools for OSIITI is that its systems will be constrained by those tools without control over their evolution. • For OSIITI, the decision on the use of open-source tools also depends on the maturity, availability, and continuing support of open-source components. • Challenges related to licenses to use open-source tools impose some restrictions on how OSIITI can use the tools. A choice of open-source tools should depend on the type of products that OSIITI wants to develop and the expertise of the development team.
Other Development tools	Does OSIITI use development tools to embed software architecture assumptions that limit architectural choices?	• Software architecture is influenced by the development technologies, such as web application frameworks, and mobile development toolkits. These technologies consider some built-in assumptions and architectural patterns. OSIITI should be informed about and conform to these assumptions and patterns. • The development technology used by OSIITI may have an indirect effect on the architecture. Developers generally use architectural options with common patterns. If OSIITI has enough experience with relational databases, for example, they might argue for this instead of a NoSQL database.

5 Summary and Conclusions

This paper aimed to perform a literature review of software architecture tools and to propose architectures for the requirement of OSIITI. We surveyed more than 50 software architecture tools. The results of this survey identified five classes, namely (a) Modeling Tools; (b) Code-Based Tools (Diagrams-As-Code); (c) [61] Automated Tools; (d) [62] Diagramming Tools; and (e) [63] Icons-Based Tools. For each class, several software tools are provided with their major features. A couple of architectures, based on layered and service-oriented patterns are proposed for OSIITI. These classes and tools are very helpful for organizations such as OSIITI that want to develop software, in both small and large-scale projects. For future research, we are going to implement the proposed architectures and make an evaluation. Architecture evaluation is the process of determining the degree to which those architectures are fit for the purpose for which they are intended. That's the role of evaluation, which is based on analyzing the alternatives in terms of several attributes, including performance, security, and usability.

References

[1] L. Bass, P. Clements, and R. Kaman. *Software Architecture in Practice*. Addison-Wesley, 4th edition, 2022.

[2] OSIITI. Organization of small industries and industrial towns of iran, 2023. URL http://isipo.ir. Last visited: 15 Jan 2023.

[3] E. Majidi, M. Alemi, and H. Rashidi. "software architecture: A survey and classification". In *Second International Conference on Communication Software and Networks*, page 454–460, 2010. doi:10.1109/ICCSN.2010.94.

[4] M. Richards and N. Ford. *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Media, 2020.

[5] J. Portillo-Rodriguez, A. Vizcaino, Ch. Ebert, and M. Piattini. Tools to support global software development processes: A survey. In *2010 5th IEEE International Conference on Global Software Engineering*, pages 13–22, 2010. doi:10.1109/ICGSE.2010.12.

[6] M. Tavakoli, L. Zhao, A. Heydari, and G. Nenadic. Extracting useful software development information from mobile application reviews: A survey of intelligent mining techniques and tools. *Expert Systems with Appli-*

[61] https://softwarearchitecture.tools/#automated-tools
[62] https://softwarearchitecture.tools/#diagramming-tools
[63] https://softwarearchitecture.tools/#diagram-and-cloud-icons

cations, 113:186–199, 2018. ISSN 0957-4174. doi:10.1016/j.eswa.2018.05.037.

[7] E. Fregnan, T. Baum, and A. Bacchelli. A survey on software coupling relations and tools, 2019. ISSN 0950-5849.

[8] M. Ozkaya and F. Erata. Understanding practitioners' challenges on software modeling: A survey. *Journal of Computer Languages*, 58:100963, 2020. ISSN 2590-1184. doi:10.1016/j.cola.2020.100963.

[9] M. Galster and D. Weyns. Empirical research in software architecture — perceptions of the community. *Journal of Systems and Software*, 202:111684, 2023. ISSN 0164-1212. doi:10.1016/j.jss.2023.111684.

[10] M.R. Behbahani Nejad and H. Rashidi. A novel architecture based on business intelligence approach to exploit big data. *Journal of Electrical and Computer Engineering Innovations (JECEI)*, 11(1):85–102, 2023. doi:10.22061/jecei.2022.8565.529.

[11] F. Tian, P. Liang, and M.A. Babar. Relationships between software architecture and source code in practice: An exploratory survey and interview. *Information and Software Technology*, 141:106705, 2022. ISSN 0950-5849. doi:10.1016/j.infsof.2021.106705.

Hassan Rashidi is a Professor in Department of Mathematics and Computer Science of Allameh Tabataba'i University. He received the B.Sc. degree in Computer Engineering and M.Sc. degree in Systems Engineering and Planning, both from the Isfahan University of Technology, Iran. He obtained a Ph.D. from the Computer Science and Electronic System Engineering department of the University of Essex, UK. His research interests include software engineering, Decision support systems, and scheduling algorithms. He has published many research papers in International conferences and Journals.

Zahra Rashidi received her B.Sc. degree in IT Engineering from Amirkabir University of Technology, Tehran, Iran, in 2014 and obtained her M.Sc. and Ph.D. degree in Computer Engineering respectively from Sharif University of Technology, Tehran, Iran, and Iran University of Science and Technology, Tehran, Iran, in 2017 and 2023. In her Ph.D. thesis, she studied the application of game-theoretic learning for caching contents in small-cell base stations.

Zeynab Rashidi received her B.Sc. and M.Sc. degrees in Computer Engineering, respectively, from Alzahra University and Amirkabir University of Technology, Tehran, Iran. She is a Ph.D. candidate in educational technology, faculty of psychology and educational Sciences, at Allameh Tabataba'i University. She has published several papers in Persian and English in journals and two books. Conducting educational workshops, teaching university courses, and collaborating on research projects are among her activities.