

A novel dynamic enterprise architecture model: leveraging MAPE-K loop and case-based reasoning for context awareness

Imane Ettahiri, Karim Doumi

Alqualsadi Research Team, ENSIAS, Mohammed V University in Rabat, Rabat, Morocco

Article Info

Article history:

Received Oct 17, 2023

Revised Nov 12, 2023

Accepted Nov 29, 2023

Keywords:

Case-based reasoning

Context-awareness

Dynamic enterprise architecture

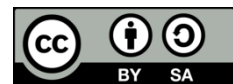
Enterprise architecture

Monitor-analyze-plan-execute-knowledge

ABSTRACT

Nowadays, enterprises are required to take the uncertainty of the environment as decisive factor of success. For this reason, Enterprises should be prepared up-stream to react dynamically to the turbulent context. Considering that enterprise architecture is a tool drawing a blueprint that gives a holistic view of the enterprise, this blueprint should be able to represent this awareness to context and implements the techniques and mechanisms to react in a dynamic manner depending on the triggers of change. In this paper, the proposed model stipulates a "context-awareness" that monitors the internal and external context, and then adapt its reaction in alignment with the prefixed goals. The operationalization of our conception is realized through the monitor-analyze-plan-execute-knowledge (MAPE-K) loop, the case-based reasoning and machine learning techniques organized and orchestrated through a global algorithm of 6 main functions to monitor, compare, analyze, plan, execute and enrich the knowledge base. The results are verified in the light of a case study that demonstrates the applicability of our proposed model.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Imane Ettahiri

Alqualsadi Research Team, ENSIAS, Mohammed V University in Rabat

Rabat, Morocco

Email: i.ettahiri@gmail.com

1. INTRODUCTION

Adapting to change and seizing emerging opportunities is crucial for the success of modern enterprises. As a result, researchers and practitioners have been developing methods, techniques, and decision-making tools that facilitate quick adaptation and response to changing circumstances. According to [1], by 2025, practitioners and researchers expect that enterprise architecture (EA) will be a well-established field that can demonstrate its ability to face the challenges of tomorrow. Therefore, the new model of EA must be able to manage deviations in working routines to avoid any inadequacies in the organization's performance due to expected or unexpected changes. The model's adaptability depends on understanding unexpected events [2]. Our research focuses on developing a dynamic enterprise architecture that meets the requirements of the current environment, characterized by its increased dynamism and perpetual evolution, while ensuring dynamic and continuous integration with its ecosystem.

Building on the valuable insights gained from our previous research works, where we exposed a comprehensive exploration of the three-layer pattern, inspired by *Dynamico* [3], a fundamental structure based on monitor-analyze-plan-execute-knowledge (MAPE-K) loop, control theory, and machine learning techniques that serve as the cornerstone for an organized, durable, and adaptive enterprise architecture behind the uncertainty of the internal and external environment. The three-layer pattern consists of the goal, adaptation, and monitoring layer. The goal layer serves as the guiding beacon for enterprise-wide strategic

objectives, steering the enterprise towards its overarching mission. The adaptation layer facilitates the identification and application of past successful cases, enabling dynamic decision-making grounded in empirical evidence. Meanwhile, the monitoring layer with the power of MAPE-K ensures real-time feedback and continuous alignment with the changing business landscape, driving timely adaptations.

Our paper focuses on the development of a dynamic enterprise architecture that meets the requirements of the current environment characterized by perpetual evolution and increased dynamism. We propose an approach that ensures dynamics in enterprise architecture by promoting self-adaptation, inspired by the well-known loop in autonomic computing MAPE-K [4]. It involves a set of algorithms aimed at monitoring the internal and external environment of the enterprise architecture and ensuring its processing via stages: analysis, comparison with the existing one, planning, execution, and ultimately, enrichment of the knowledge base. Based on machine learning techniques, our proposed model ensures continuous integration with the ecosystem and drives strategic decision-making, promoting continuous development to reach high-level goals [5].

Our paper consists of 7 sections. In section 2, we present related works that deal with dynamic enterprise architectures and our fundamental theoretical concepts, MAPE-K and case-based reasoning (CBR). In section 3, we explain the research methods we used in our paper. In section 4, we present our proposed model to dynamic enterprise architecture. To validate the effectiveness of our approach, we present a use case that focuses on “dynamic business continuity planning during a pandemic disease like COVID-19” in the section 5 “experiment”. This use case illustrates how enterprises can quickly adapt their operations to ensure business continuity amidst unprecedented challenges. In section 6, we discuss and analysis our finding and proposed model and evaluate it in comparison with related works. In the final section, we conclude with a summary of our main findings and future perspectives.

2. RELATED WORKS

2.1. Dynamic enterprise architecture

The English dictionary oxford learners defines dynamic as a property (of a system, process, or relationship) that is always changing and progressing. It is the inverse of static and is extensively investigated in fields including sociology, bacteriology, mechanics, statistics, geophysics and hydrology. Because dynamic is still a difficult paradigm [6], we attempted to investigate various aspects of dynamics in enterprise architecture in our earlier study [7]. As observed, the dynamic aspect of enterprise architecture varies in accordance with the decomposition prism used: dynamic capabilities view, viewpoint view (with the Zachman meaning), service view, dynamic design layer view and agility-centric view. This dynamic characteristic pervades all scales (holistic, dynamic components, inter-enterprise, intra-enterprise), and during the EA's several action phases (analyzing, modeling, designing, planning, implementation, and measurement). The benefits of each of the precited approaches/studies vary between the consistency of the static element's stability and the flexibility and agility of the dynamic aspect in EA. Explanatory studies aid in the development of an improved comprehension of complicated EA reality for trustworthy representation, as well as a deeper comprehension of dynamic capabilities of EA for organizational benefits. We categorized the six studied approach on: dynamic service-oriented approach, dynamic pattern-oriented approach, dynamic capabilities-oriented approach, chaos theory approach, and agility centric approach [7]. In our works, we attempted to create a model as accurate as possible, to gather the largest number of benefits presented in previous work.

2.2. Case-based reasoning

According to Leake [8], CBR is a reasoning by remembering method. It is a technology-independent process employed by humans and simulated in information systems [9]. CBR is defined [10] as both the ways individuals use cases to solve problems and the methods, we can teach machines to use. CBR can use specialized knowledge gathered from previously solved concrete problem scenarios (cases) to draw on. Finding a similar previous example and applying the solution to the new problem circumstance solves the new problem [11]. As previously stated, the case-based reasoning life cycle consists of the following steps: i) Using a similarity algorithm, extract the most similar instance(s) from the case base, which comprises earlier cases, based on the description of the present scenario provided as query; ii) Adapt the retrieved lesson to the new scenario, which becomes part of a new case; repurpose the lesson from the retrieved case(s) as the proposed remedy for the new predicament; and iii) Revise the new case after analyzing it in the new situation.

CBR has also been used for EA management in several recent works. For example, Zhang *et al.* [12] proposes a CBR-based approach for selecting EA patterns in the context of digital transformation [13] proposes a CBR-based approach for recommending EA changes to improve the security of enterprise architectures [14] proposes a CBR-based approach for managing the complexity of enterprise architectures.

These works demonstrate the potential of CBR for improving the efficiency and effectiveness of EA management.

2.2.1. Monitor-analyze-plan-execute-knowledge feedback loop

This research focuses on architecture-based self-adaptive systems [15]–[17]. A self-adaptive system consists of a managed system that is controllable and adaptable, as well as the management system that accomplishes the managed system's adaptations. The managed system functions in an uncontrollable environment. The management system implements a feedback loop that includes four critical functions: MAPE-K, or MAPE in short, stands for monitor, analyze, plan then execute that shares knowledge [18]. The monitor keeps track of the system to manage and its environment, and it keeps the knowledge up to current. The analyzer evaluates the requirement for adaptation using current knowledge, which may include rigorous analysis techniques [19]–[21] or simulations of runtime models [22]. If adaptation is required, such analysis could utilize rigorous procedures to give assurances for the controlled system. These alternate configurations are known as adaption choices. The planner then chooses the optimal alternative based on the adaptation goals and creates a plan to transition the system from its present configuration to the new configuration. Finally, the plan's adaption activities are carried out by the executor. It is vital to note that MAPE gives a reference model that outlines the fundamental operations of a managing system and their relationships. A reel use case architecture connects functions to the components, this relationship could be one-to-one or any other mapping, such as mapping the analytical and planning functions to a single integrated decision-making component. In recent years, there has been growing interest in using the MAPE-K loop for EA management. For example, Zhu *et al.* [23] proposes a MAPE-K-based approach for managing the self-adaptability of EA models. Liu *et al.* [24] proposes a MAPE-K-based approach for managing the deployment and maintenance of EA solutions in a cloud environment. Wang *et al.* [25] proposes a MAPE-K-based approach for managing the evolution of EA models in a DevOps environment. These works demonstrate the potential of the MAPE-K loop for improving the agility and adaptability of enterprise architectures.

3. RESEARCH METHOD

We used the information systems design science methodology in our study. When conducting research in information science, a variety of strategies, concepts, techniques, and perspectives—including interpretative, critical, and complementary ones—are referred to as design science [26]. To advance both the state of practice and the body of research knowledge currently in existence, the design science technique is frequently used to develop and assess an artifact and/or a design theory [27]. We used Peffer *et al.* design science research methodology (DSRM) process model [28] to organize our research endeavor. Six main actions make up this sequential process model: problem identification and motivation, solution objectives formulation, design and development, demonstration, evaluation, and communication, as shown in Figure 1.

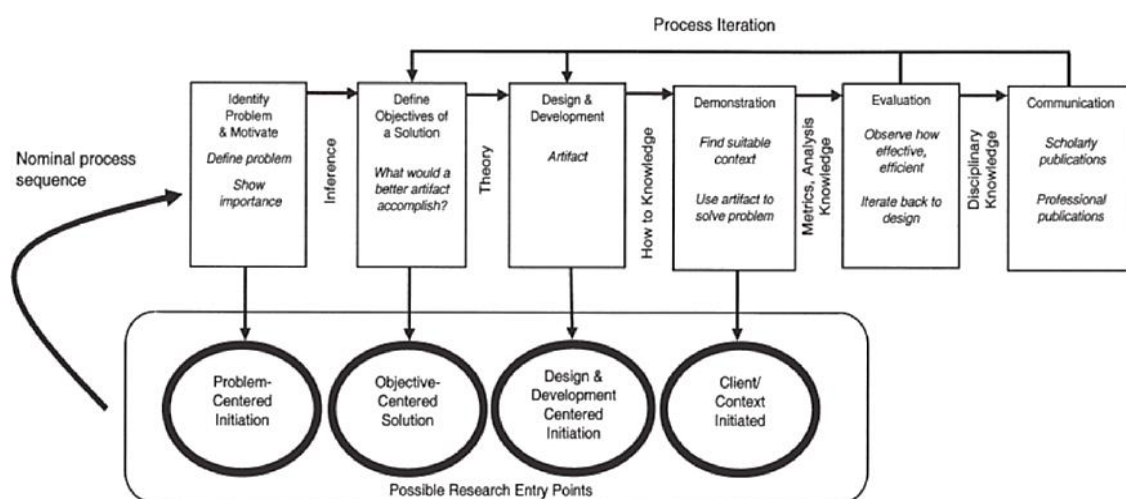


Figure 1. The process model of DSRM [28]

The process of solving a research problem involves several steps. The first step is to identify the problem and explain why it is important to solve it. The second step is to determine the objectives of the

solution based on the problem specification. These objectives can be either quantitative or qualitative. The third step is to design and develop an artifact that can be a construct, model, method, or instantiation. This requires a good understanding of the theory that can be used to find a solution. The fourth step is to demonstrate the efficacy and efficiency of the artifact by using it to solve the problem. This can be done through simulations, proofs, case studies, or experimentation. The fifth step is to evaluate the created artifact by observing and measuring its effectiveness in solving the problem. This requires knowledge of relevant metrics and analysis techniques. The final step is to communicate the problem, the artifact, and its effectiveness to technical audiences and researchers for better comprehension and evaluation. Having defined and formulated the problem of our research paper, we proceed with the following:

- a. Objectives and solution: The objective was to develop a solution that will identify any new trigger of change, compare it with the existing cases in the knowledge base, analyze it, calculate the similarity, and then propose a new plan based on the existing cases or/and the experts' recommendations, and finally execute it and enrich the knowledge base with the new information.
- b. Design and development: The solution we proposed was a model based on a 6-steps algorithm. Combining the MAPE-K feedback loop and the CBR algorithm.
- c. Demonstration and evaluation: We tested the applicability of our model with a use case of dynamic business continuity planning during a pandemic diseases like coronavirus disease (COVID-19).
- d. Communication: the communication is via publishing our results in our previous participations in conferences and papers.

4. PROPOSED MODEL

This section delves into the core of the proposed model, unveiling its macro-architecture in sub-section 4.1. Subsequently, sub-sections 4.2 to 4.7 detail the six crucial steps of the process: monitoring, comparing, analyzing, planning, acting, and ultimately, updating the knowledge base. This comprehensive breakdown seeks providing a clear understanding of the model's inner workings and its systematic approach to achieve its objectives.

4.1. Macro-architecture of the proposed model

Building upon our prior research exploring essential criteria for a dynamic enterprise architecture model, we present a macro-architecture in Figure 2 that adheres to the main architectural principles [29]: flexibility, agility, expressiveness, extensibility, modularity and reuse, durability and prediction, context awareness, and intelligent reaction. This architecture leverages lived experiences, feedback, experiments, simulations, and expert opinions to ensure continuous adaptation and responsiveness to evolving needs. The proposed macro-architecture represents an encapsulation of theoretical foundations and practical insights, positioning it as an adaptive model for dynamic enterprise architecture.

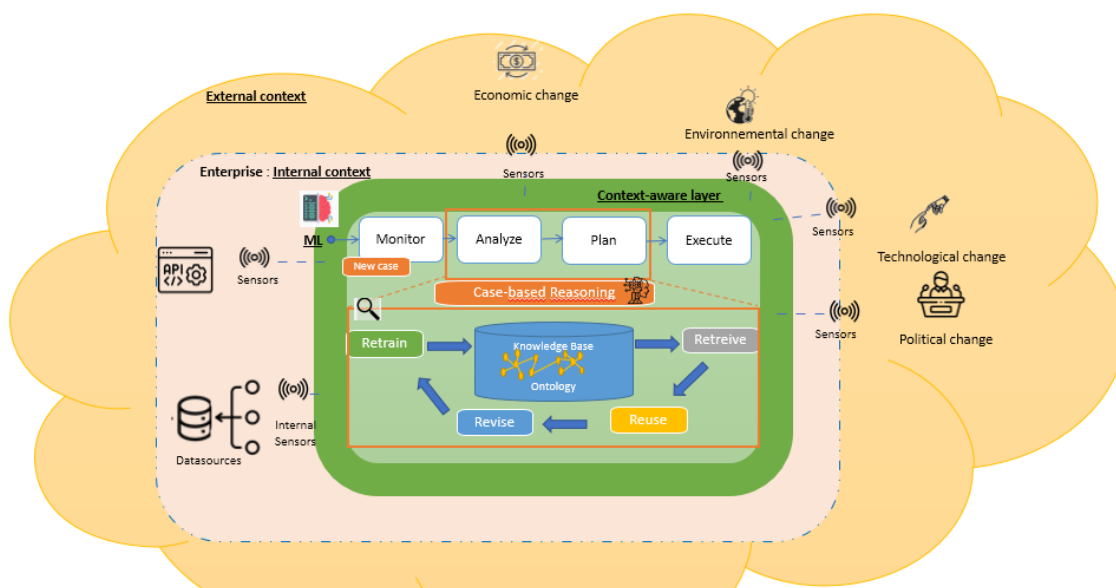


Figure 2. The macro-architecture of the proposed model

It is a process composed of six steps, as presented in Figure 3, and translated to an algorithm with six essential functions: monitor, compare, analyze, plan, act, and finally update the knowledge base. The six steps combine the two main concepts: MAPE-k loop and case-based reasoning steps. Then, the global algorithm *DynamicEA_MAPEK_CBR()* is presented with the six main functions.

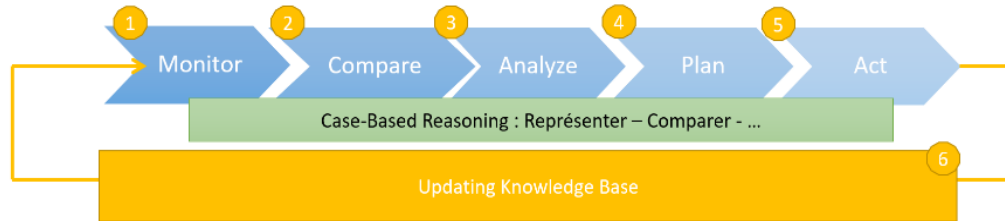


Figure 3. The 6-steps proposed process

Algorithm 1. *DynamicEA_MAPEK_CBR()*:

```

1: context_data ← monitor()
2: similar_cases ← retrieve_similar_cases(context_data)
3: analyzed_cases ← analyze_cases(similar_cases, context_data)
4: adaptation_actions ← plan_actions(analyzed_cases, context_data)
5: execute_actions(adaptation_actions)
6: update_knowledge_base(analyzed_cases)
  
```

4.2. Monitoring

This sub-section focuses on the critical element of monitoring both internal and external contexts. To effectively achieve this, we propose an algorithm that leverages machine learning techniques. This approach enables the dynamic model to stay abreast of evolving internal and external factors, ensuring its continuous alignment with the ever-changing business landscape. By integrating machine learning algorithms into the monitoring process, the model gains valuable insights into the dynamic environment, paving the way for proactive responses and informed decision-making. The proposed algorithm is presented below:

Algorithm 2. *monitor()*

```

1: Result: context_data
2: #Initialize a data structure to store context data
3: context_data ← empty_data_structure
4: for i ← 0 to i < List_type_context_data.Length - 1
5:     context_type_data() ← get_type_context_data
6:     context_data.add(context_type_data)
7: end for
  
```

The objective of this algorithm *monitor()* is to get the contextual data for each of the types of predefined and stored contexts as a dynamic list. The output is a data structure of the contextual data collected. For example we can have the above functions to constitute our context data: *get_business_process_data()*, *get_environmental_data()*, *get techno_infrastructure_data()*, *get_operational_metrics_data()*, ... The implementation of the *get_contexttype_data()* function, can be carried out via interactions with Application programming interface (APIs), data sources, databases and other business systems, or even from unstructured data (newspapers and twit) to collect relevant information for the data context. For each type of *context_data*, the implementation of its getter function will differ, with a data quality mechanism to ensure the quality of the data collected, considering that it is the entry point of our system and the basis of the entire chain of decision, analysis, and adaptation in our dynamic enterprise architecture.

Machine learning (ML) techniques are useful in this case, to better train our model to intercept data and subsequently map it with our ontology "formalism understood, agreed and interpreted by our system". natural language processing (NLP) may also be relevant for the use case of interception of data from unstructured sources. The next example, is for the function *get_business_process_data()* using NLP techniques to map the result of an unstructured data with a predefined ontology:

4.3. Comparing

The second step of the proposed process is to compare the collected contextual data to the previous cases stored in the knowledge base. This algorithm *compare()*, retrieves similar cases by taking as input the

context_data; the result of the monitoring algorithm, and returns a list of similar cases. This algorithm is based on two main functions: *get_control_signals()* and *calculate_similarities()*.

Algorithm 2.1. *get_business_process_data()*

```

1: Result: ontology_mapping
2: # Step 1: Retrieve unstructured text data from your data source
3: raw_text_data ← retrieve_unstructured_text_data()
4: # Step 2: Preprocess the raw text data (e.g., tokenization,
5: #stopword removal, stemming)
6: preprocessed_data ← preprocess_text_data(raw_text_data)
7: # Step 3: Use NLP techniques to extract relevant information
8: extracted_information ← extract_info_with_nlp(preprocessed_data)
9: # Step 4: Use ML classifier to categorize the extracted #information
10: ml_model ← train_ml_model(training_data)
11: categorized_data ← classify_with_ml_model(extracted_information, ml_model)
12: # Step 5: Map categorized data to ontology classes
13: ontology_mapping ← map_to_ontology_classes(categorized_data)

```

Algorithm 3. *compare(context_data)*

```

1: Result: similar_cases
2: # Initialize an empty list to store similar cases
3: similar_cases ← EMPTY_LIST
4: # Get control signals from the current context
5: control_signals ← get_control_signals(context_data)
6: for each case in case_base:
7:     similarity_score ← calculate_similarity(case, context_data, control_signals)
8:     if similarity_score >= similarity_threshold then
9:         # Apply a threshold to filter relevant cases
10:        similar_cases.ADD(case)
11:    end if
12: end for

```

For the details of the two main functions: *get_control_signals(context_data)*

Algorithm 3.1. *get_control_signals(context_data)*

```

1: Result: control_signals
2: # Initialize an empty dictionary to store control signals
3: control_signals ← EMPTY_DICT
4: # Retrieve and calculate the control signals based on the current context data
5: control_signals["urgency"] ← calculate_urgency(context_data);
6: control_signals["resource_availability"] ← calculate_resource_availability(context_data);
7: control_signals["priority_level"] ← calculate_priority_level(context_data);
8: # more control signals could be added based depending on the specific use case

```

It is worthwhile to mention that in the context of our dynamic enterprise architecture, sensitive to its internal and external context, control signals represent dynamic factors that influence decision-making and the adaptation of the enterprise. These signals help to understand the current internal state of the company as well as its external environment to make informed decisions. Table 1 presents are some examples of control signals for enterprise architecture:

For the second function of calculating similarities:

This algorithm is an implementation of the mathematical function already presented in our paper [30] for calculating similarities, where we considered the vectorial representation (n attribute for each case):

- $A(a_1, a_2, a_3, \dots, a_n)$: input case knowledge (new problem),
- $B(b_1, b_2, b_3, \dots, b_n)$: an existing case problem in the case base,
- $W(w_1, w_2, \dots, w_n)$: is weight coefficient that comes from the calculation of the impact of control signals.

$$0 \leq w_i \leq 1 \text{ with } (i = 1, 2, 3, \dots, n) \text{ and } \sum w_i = 1$$

- $Sim(a_i, b_i)$: represents a function of similarity between the i^{th} attribute of B and the i^{th} attribute of A with real number value in interval (0,1).

Assuming that each of the attribute is given the weight w_i , the similarity degree between the input case and the targeted case is given by $Sim(A, B)$, which is formulated by:

$$Sim(A, B) = \sum w_i \times Sim(a_i, b_i)$$

Table 1. Classification of context sensitive control signals

Type of context sensitive Control signals	Examples of context sensitive Control signals per type
Internal	Performance metrics
	Resource usage
	Capacity constraints
	Operational risks
	Market trend
	Legislative change
External	Economic indicators
	Socio-economic factors
	Socio-cultural factors
	Technological advancement
	Environmental factors
	Political factors
	Partnership opportunities

Algorithm 3.2. calculate_similarity (case, context_data, control_signals)

```

1: Result: total_similarity_score
2: total_similarity_score ← 0
3: # Step 1: Calculate similarity scores for each attribute in the case
4: for each attribute in case:
5:     if attribute in context_data then
6:         similarity_score ←
           calculate_attribute_similarity(attribute_value_in_case,
           context_data.get(attribute))
7:         total_similarity_score ← total_similarity_score+similarity_score
8:     end if
9: end for
10: # Step 2: Adjust similarity based on control signals
11: for each control_signal in control_signals:
12:     if control_signal in case then
13:         control_signal_weight ←
           calculate_control_signal_weight(control_signals.get(control_signal))
14:         total_similarity_score ← total_similarity_score*control_signal_weight
15:     end if
16: end for

```

4.4. Analyzing

The third step is about analyzing the similar cases. The proposed algorithm is to analyze the retrieved similar cases and current context to identify patterns, trends, and potential solutions. Then, the detail of the algorithm:

Algorithm 4, "analyze", is a process that evaluates the relevance of a set of "similar_cases" based on contextual information provided by "context_data." Its goal is to determine the suitability of these cases within the current context. The primary outcome of this algorithm is a list called "analyzed_cases_result," which will contain information about the relevance of each case. To achieve this, the algorithm follows the steps mentioned below:

- a. Initialization: Start by initializing an empty list called "analyzed_cases_result".
- b. Empty cases handling: If there are no "similar_cases" available (i.e., the "similar_cases" list is empty), the algorithm returns an empty list, indicating that there's nothing to analyze in this context.
- c. Iterating over similar cases: For each "case" within the "similar_cases" list, the algorithm proceeds with the following steps:
 - Case relevance evaluation: The algorithm calculates a "relevance_score" for the current case by invoking the "analyze_case_relevance" function. This score reflects how well the case aligns with the context provided in "context_data". The exact calculation method for this relevance score is not detailed in this algorithm but is delegated to the "analyze_case_relevance" function.
 - Storing analyzed case information: The "analyzed_case" object is created to hold both the case itself and its associated "relevance_score". This information is stored as a dictionary in the "analyzed_cases_result" list.
 - Updating the result list: The "analyzed_case" is added to the "analyzed_cases_result" list for further processing.
- d. Completion: Once all similar cases have been analyzed, and their relevance scores have been determined, the algorithm returns the "analyzed_cases_result" list. This list contains information about the relevance of each case in the context.

The algorithm references an external function called "*analyze_case_relevance*" (defined elsewhere) for calculating the relevance score of a given case based on the provided "*monitor_data*" and other context-related information. The specific implementation of this function may involve methods like similarity metrics, context evaluation, or other relevant calculations.

Algorithm 4 serves as a fundamental step in determining the contextual relevance of cases, which is crucial for various decision-making processes, such as case-based reasoning systems, recommendation engines, or context-aware applications. The specific logic for assessing relevance is encapsulated within the "*analyze_case_relevance*" function, enabling flexibility and adaptability to different contexts and domains. This allows the algorithm to effectively assess the relevance of cases across various scenarios, paving the way for informed decision-making.

Algorithm 4. Analyze (*similar_cases*, *context_data*):

```

1: Result: analyzed_cases_result
2: # Initialize a list to store the analyzed cases
3: analyzed_cases_result = []
4: # Return an empty list when no similar cases are available
5: if is_empty(similar_cases) then
6:     return analyzed_cases_result
7: end if
8: for case in similar_cases:
9:     # Analyze - Evaluate the relevance of each case based on context
10:    relevance_score = analyze_case_relevance(case, monitor_data)
11:    # Store the relevance score and the case for further processing
12:    analyzed_case = {'case': case, 'relevance_score': relevance_score}
13:    analyzed_cases_result.append(analyzed_case)
14: end for
15: return analyzed_cases_result
16: # Define a function to analyze the relevance of a case based on context
17: function analyze_case_relevance(case, monitor_data):
18: # The implementation to calculate the relevance of the case based on context data
19: # Consider using similarity metrics and context evaluation
20: # Return a relevance score indicating the case's suitability for the current context
21: end

```

4.5. Planning

In this step of planning, the used algorithm is the *plan_actions(analyzed_cases, context_data)*. This 5th Algorithm is a critical component for enterprise architecture (EA) evaluation and adaptation. It provides a systematic approach to recommend and select actions based on relevant cases and contextual data analysis. The algorithm efficiently sifts through a list of analyzed cases and recommends actions based on the analysis. This process streamlines decision-making, which is necessary in a dynamic and complex environment. It is a well-structured algorithm to handle scenarios where no recommended actions are available or when none of the analyzed cases yield suitable recommendations. The inclusion of "*take_default_actions_or_handling()*" ensures that there's a backup plan or default action strategy in place, a vital feature for maintaining the stability of the EA. This algorithm ensures contextual evaluation, actually, the step to evaluate the recommended actions based on context data, control signals, or other criteria is a critical aspect of the algorithm. It reflects the ability of the EA to make informed decisions that align with the current context. This feature is crucial for ensuring the adaptiveness and context-awareness of the architecture.

- Structured approach: The algorithm's clear structure and use of conditional statements make it highly comprehensible and maintainable. It adheres to good coding practices and facilitates easy troubleshooting and further development.
- Adaptability: The algorithm's adaptability is apparent in its ability to select actions based on the evaluation process. This adaptability is an essential feature for the EA's capability to respond to changing circumstances and fulfill its objectives effectively.
- User-centered approach: The algorithm considers the possibility of no suitable recommendations, which could arise in real-world scenarios. By providing a mechanism to handle such situations, it maintains a user-centered approach to EA. It ensures that the architecture can handle exceptions and respond in a way that serves the organization's best interests.

In summary, Algorithm 5: *plan_actions* is a valuable component of EA evaluation and adaptation. It offers a systematic approach to recommend, evaluate, and select actions, promoting a responsive and adaptable EA. Its error-handling capabilities and user-centered approach ensure the reliability and robustness of the architecture in the face of dynamic challenges and changing contexts.

Algorithm 5. plan_actions(analyzed_cases, context_data)

```

1: Result: selected_actions
2: # Initialize an empty list to store the recommended actions
3: recommended_actions ← EMPTY_LIST
4: for each analyzed_case in analyzed_cases:
5:     recommended_action ← retrieve_recommended_action(analyzed_case)
6:     if recommended_action is not NULL then
7:         recommended_actions.append(recommended_action)
8:     end if
9: end for
10: if recommended_actions is not EMPTY then
11:     # Evaluate the recommended actions based on context data,
12:     # control signals, or other criteria
13:     selected_actions ← evaluate_actions(recommended_actions, context_data)
14:     return selected_actions
15: else:
16:     # Handle the case where no recommended actions are #available or none
17:     # of the analyzed cases led to suitable recommendations.
18:     # Take appropriate default actions or handling.
19:     take_default_actions_or_handling()
20:     return default_actions
21: end if

```

4.6. Acting

The provided algorithm is a crucial step in the EA process, where adaptation actions are executed. The algorithm plays an essential role in making sure that the EA can remain responsive and adaptable to changes in its environment. It is important to mention some comments on the algorithm:

- **Efficient execution:** The algorithm efficiently iterates through a list of adaptation actions and executes each one sequentially. This approach allows for systematic and controlled adaptation, which is crucial for maintaining the integrity of the EA.
- **Error handling:** The algorithm provides optional steps for verifying the successful execution of actions and handling errors or exceptions. This robust error-handling mechanism is essential in ensuring that the EA can respond gracefully to unexpected issues during execution.
- **Control signal and context data updates:** The optional steps to update control signals and context data based on the executed actions are noteworthy. These updates can have a profound impact on the adaptiveness and context-awareness of the EA. By incorporating these steps, the algorithm ensures that the architecture is well-informed and can adapt proactively.
- **Triggering further actions:** The algorithm's optional step to trigger further actions or processes based on the executed actions adds another layer of sophistication. This ability to initiate subsequent actions based on the results of previous actions reflects a dynamic and responsive approach to EA.
- **Result reporting:** The final step of returning or notifying the execution result is crucial for feedback and accountability. It allows for tracking the success and outcomes of the adaptation process, which is essential for organizational learning and improvement.

Overall, Algorithm 6 embodies the essence of a dynamic and context-aware enterprise architecture. It not only executes actions but also provides mechanisms for error handling, data updates, and proactive response. In the ever-changing landscape of modern organizations, such an algorithm is invaluable for ensuring that the architecture remains in sync with the evolving needs and challenges of the business.

Algorithm 6. execute_actions (adaptation_actions)

```

1: Result: selected_actions
2: for each action in adaptation_actions:
3:     EXECUTE action
4:     # Optionally, verify the successful execution of actions and handle any errors
5:     # or exceptions
6:     # Optionally, update control signals and context data based on the executed
7:     # actions
8:     UPDATE_CONTROL_SIGNALS_AND_CONTEXT()
9:     # Optionally, trigger further actions or processes based on the executed actions
10:    TRIGGER_FURTHER_ACTIONS()
11:    # Return or notify the result of the action execution process
12:    RETURN_OR_NOTIFY_EXECUTION_RESULT()
13: end for

```

4.7. Updating knowledge base

This last step is for updating and enriching the knowledge base with the analyzed cases. This 7th algorithm, "*update_knowledge_base*", is responsible for improving the knowledge base by incorporating

new information from a set of "*analyzed_cases*". It goes through each case, evaluates its outcome, and if a valid outcome exists, it updates the knowledge base with the analyzed case and its corresponding outcome. This algorithm plays a crucial role in maintaining and refining the knowledge base, enabling it to become more accurate and comprehensive over time. By systematically integrating new data, it allows the knowledge base to evolve and adapt to changing circumstances, which is essential for decision support systems, machine learning, or any application that relies on a knowledge base for informed decision-making. While this algorithm demonstrates a basic structure for updating a knowledge base, it does not specify the exact implementation of the "UPDATE *knowledge_base* WITH" function, which may vary depending on the underlying system or database being used. It also assumes that the "*get_case_outcome*" function retrieves the outcome of each case accurately. The reliability and effectiveness of this algorithm depend on the quality of the analyzed cases, the accuracy of the outcome retrieval process, and the soundness of the knowledge base update mechanism. Proper error handling and data validation are crucial for robust operation.

Algorithm 7. update_knowledge_base(analyzed_cases)

```

1: Result: knowledge_base
2:   for each analyzed_case in analyzed_cases:
3:     case_outcome ← get_case_outcome(analyzed_case)
4:     # Retrieve the outcome of the analyzed case
5:     if case_outcome is not NULL then:
6:       UPDATE knowledge_base WITH (analyzed_case, case_outcome)
7:     end if
8:   end for

```

5. EXPERIMENT

In this section, a use case is presented to experiment, and then demonstrate how the algorithms of MAPE-K, CBR, and the three-layer pattern (goal, adaptation, and monitoring layers) work together in a dynamic enterprise architecture context. The proposed use case is about dynamic business continuity planning during a pandemic diseases like COVID-19, the proposed scenario is as follows: in response to the pandemic, an organization needs to adapt its business operations to ensure continuity while safeguarding the health and safety of its employees and customers. The dynamic enterprise architecture model will be used to dynamically adjust business processes, technology infrastructure, and employee workflows during the pandemic. The goal layer defines the high-level strategic goal: "Ensure business continuity and resilience during the COVID-19 pandemic". The specific objectives are consequently: enable remote work, maintain essential services, and ensure customer support amid restrictions and uncertainties.

The monitoring layer, that is a real-time monitoring, tracks COVID-19 cases and restrictions in various regions of operation, monitors employee health and well-being, monitors customer demand and feedback. For the adaptation layer, and based on the analysis, the organization decides to implement a remote work policy for employees where feasible, it invests in cloud-based collaboration tools and virtual private networks (VPNs) for secure remote access, the organization also introduces contactless delivery options and enhances its online customer support channels. Then, the six-step process based CBR with MAPE-K analysis:

Step 1: Monitor, real-time COVID-19 cases, regulations, employee health, and customer feedback.

Step 2: Compare, this step focuses on comparing the available solutions and adaptation strategies to determine which one is most suitable for the organization's specific needs.

Step 3: Analyze past business continuity cases for relevance based on current monitoring and evaluate successful adaptations made during the pandemic by other enterprises.

Step 4: Plan, this step determinate the best business continuity strategies based on the relevance score and analysis of successful adaptations by other enterprises, and plan for remote work setup, digitalization of services and enhanced customer support.

Step 5: Execute, the organization executes the adaptation plan, enabling remote work for employees, deploying digital tools, and implementing contactless delivery.

Step 6: Enrich the knowledge base, after each adaptation, update the knowledge base with the outcomes and lessons learned during the pandemic, and store successful business continuity patterns for future reference and to prepare for potential future crises.

The ontological representation provides a structured and formalized framework for representing the key concepts, relationships, and actions involved in dynamic business continuity planning [31]. By adopting this approach, organizations can enhance their resilience and agility, enabling them to navigate through challenges like the COVID-19 pandemic effectively. As a result, decision-makers can make informed and data-driven choices, ensuring the survival and success of the enterprise in times of crisis. However, to realize the potential of this model, further research and implementation are necessary to validate its efficacy and

suitability across various industries and scenarios. We expose in Figure 4, an ontological representation of our use case with the OWL language.

The dynamic enterprise architecture model helps the organization dynamically adjust its operations to continue activities during the COVID-19 pandemic. By learning from successful adaptations made by other enterprises, the organization can make informed decisions for its own business continuity planning. The three-layer pattern of goal, adaptation, and monitoring ensures a systematic approach to maintaining business continuity and resilience during challenging times. By utilizing the dynamic enterprise architecture model with the MAPE-K architecture, CBR, and the three-layer pattern, the organization can adapt its operations swiftly during the COVID-19 pandemic. The model allows the organization to monitor real-time data, analyze successful cases from other enterprises, plan and execute effective business continuity strategies, and continuously enrich its knowledge base for future resilience and preparedness.

```

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix covid: <http://example.com/ontology/covid#>.

# Monitor Step
covid:COVIDData rdf:type owl:Class.
covid:GovernmentRegulations rdf:type owl:Class.
covid:EmployeeHealthData rdf:type owl:Class.
covid:CustomerFeedbackData rdf:type owl:Class.
covid:MarketTrends rdf:type owl:Class.

covid:hasCOVIDCaseData rdf:type owl:ObjectProperty.
covid:hasGovernmentRegulations rdf:type owl:ObjectProperty.
covid:hasEmployeeHealthData rdf:type owl:ObjectProperty.
covid:hasCustomerFeedbackData rdf:type owl:ObjectProperty.
covid:hasMarketTrends rdf:type owl:ObjectProperty.

# Compare Step
covid:HistoricalData rdf:type owl:Class.
covid:HealthEmergencies rdf:type owl:Class.

covid:hasHistoricalData rdf:type owl:ObjectProperty.
covid:hasHealthEmergencies rdf:type owl:ObjectProperty.

# Analyze Step
covid:RiskAnalysis rdf:type owl:Class.
covid:BusinessImpact rdf:type owl:Class.

covid:hasRiskAnalysis rdf:type owl:ObjectProperty.
covid:hasBusinessImpact rdf:type owl:ObjectProperty.

# Plan Step
covid:BusinessContinuityPlan rdf:type owl:Class.
covid:AdaptationStrategies rdf:type owl:Class.

covid:hasBusinessContinuityPlan rdf:type owl:ObjectProperty.
covid:hasAdaptationStrategies rdf:type owl:ObjectProperty.

# Execute Step
covid:DynamicExecution rdf:type owl:Class.
covid:ContextAwareResponse rdf:type owl:Class.

covid:hasDynamicExecution rdf:type owl:ObjectProperty.
covid:hasContextAwareResponse rdf:type owl:ObjectProperty.

# Enrich Knowledge Base Step
covid:KnowledgeBase rdf:type owl:Class.
covid:LessonsLearned rdf:type owl:Class.

covid:hasKnowledgeBase rdf:type owl:ObjectProperty.
covid:hasLessonsLearned rdf:type owl:ObjectProperty.

```

Figure 4. OWL representation for the ontology applied to each step of the process

6. DISCUSSION AND ANALYSIS

In this section, we discuss and analyze the proposed dynamic enterprise architecture model, applied in a practical use case in section 5. Our algorithm integrates the MAPE-K loop with the EA algorithm: “case-based reasoning” and includes machine learning techniques for context awareness to enrich the knowledge base and make it more robust and responsive. The model comprises six key steps:

- Monitoring the external and internal context of the enterprise through machine-learning techniques, with the ultimate objective of creating a context-aware model that can intercept changes at the appropriate moment.
- Comparing cases with existing knowledge.
- Analyzing the cases.
- Planning the necessary transformations.
- Executing the planned transformations.
- Updating the knowledge base.

The strength of our model increases proportionally with the enrichment of the knowledge base around which our approach revolves. This enrichment is ensured by simulations, expert recommendations, and sharing with other knowledge bases. This model enables dynamic enterprise architecture guided by flexibility, agility, adaptiveness, modularity, and expressiveness, which aligns with the proposed architectural principles [7].

To position our approach among other studies and approaches that deal with the dynamic aspect of enterprise architecture, and to evaluate our proposal in good practices, we must take into account the broader definition of evaluation indicators in the context of EA. Unlike software or service-oriented architecture evaluation, EA evaluation indicators can be defined from various perspectives. Building on our previous works and the insights provided by Mirsalari and Ranjbarfard [32], we have developed an EA evaluation model that is based on five criteria, each of which is aligned with a specific architectural principle. At the outset of our work, we compiled a list of architectural principles that our solution must adhere to [7]. An architectural principle is a statement that prescriptively specifies a property of a design artifact that is essential to meet its requirements [33]. These principles are derived from [34] and [35] and are used to describe our system architecture.

During the selection phase, we compared the architectural principles with the advantages we identified in our previous research on the dynamic aspects of enterprise architecture. We created a comprehensive list by combining the advantages of various approaches and studies related to the dynamic aspects of EA. These benefits include low coupling, high cohesion, coherence, flexibility, agility, pragmatism, semantic rigor for successful communication and documentation, reactivity, innovation, tools to direct the transformation effort towards predictable and beneficial results, a deep understanding to delineate dynamic EA capabilities to bring organizational benefits, and more. We then matched these benefits with the main criteria to ensure the most comprehensive model possible. The criteria we used were flexibility and adaptability, expressiveness, modularity and reuse, extensibility, durability, and predictability.

We have reviewed various approaches and studies that explore the dynamic aspects of enterprise architecture, using five criteria to evaluate them. Our evaluation is presented in Table 2, which includes the first six approaches/studies that were evaluated in previous research [7]. Additionally, we have provided our evaluation of the proposed approach in the last column.

Table 2. Evaluation of approaches/studies of dynamic enterprise architecture

Criteria	Approaches/Studies of enterprise architecture dealing with dynamic aspect [7]						
	Dynamic service oriented approach [36]	Dynamic pattern-oriented approach [37]	Dynamic capabilities oriented approach [38]–[40]	Chaos-theory oriented approach [6]	Dynamic metamodel oriented approach [41]	Agility-centric oriented approach [42]–[44]	Our proposed approach
Flexibility and Adaptability	X		X		X	X	X
Expressiveness				X	X	X	X
Extensibility		X					X
Modularity and Reuse	X						X
Durability and prediction		X					X

Our initial goal was to make the EA more flexible by dynamically responding to various triggers of changes. Integrating the MAPE-K loop with case-based reasoning helps to connect real-time monitoring with informed decision-making. By using machine learning and the dynamic case-based approach, the EA can evolve as the organizational landscape changes. The EA becomes agile enough and adaptive enough to respond to contextual shifts and new challenges, which is in line with the criteria of flexibility and adaptability. The model's design resonates with two criteria-modularity and reuse. Breaking down the process into discrete steps facilitates modularity, allowing each element to be refined and improved independently. In addition, the EA model's context awareness enhances adaptiveness, so it can recognize and react to various triggers of change, ensuring that the architecture remains in sync with the evolving needs of the organization. The model also addresses expressiveness, this criterion emphasizes clear and effective communication. By enriching the knowledge base through simulations, expert recommendations, and secure inter-enterprise knowledge sharing, the EA becomes more expressive and fosters improved communication and documentation of complex architectural transformations.

The proposed EA model is a significant step forward in the realm of enterprise architecture. It embodies the architectural principles of flexibility, agility, adaptiveness, modularity, durability, prediction [17], [44] and expressiveness, offering a dynamic, context-aware, and adaptable approach to EA. It enriches

the knowledge base and aligns the architecture with real-world challenges and changes, offering a promising avenue for contemporary organizations seeking to thrive in dynamic environments.

7. CONCLUSION

In this paper, we have explored the concept of dynamic EA and proposed a novel approach to achieve adaptability, responsiveness, and resilience in enterprise systems. The key components of our approach include a six-step process combining the MAPE-K architecture and CBR. Through this integration, the proposed dynamic enterprise architecture model facilitates continuous adaptation to changing business requirements, technological advancements, and external environmental conditions. This model enables enterprises to proactively respond to dynamic challenges and seize opportunities, ensuring sustainable growth and competitive advantage.

Our proposed dynamic enterprise architecture model consists of six key steps, namely monitoring, comparing, analyzing, planning, acting and updating the knowledge base. The monitoring step gathers real-time data and contextual information from internal and external sources. Comparing and analyzing leverage CBR techniques to draw insights from past successful cases and adapt those solutions to the current situation. By learning from historical data and patterns, our model becomes increasingly intelligent in making decisions and anticipating future scenarios. The Planning and Acting steps then facilitate the adaptation process by translating analysis results into concrete actions. The system dynamically adjusts business processes, allocates resources, and optimizes technology infrastructure to align with strategic goals and objectives. This adaptability empowers enterprises to meet changing customer demands, regulatory requirements, and market dynamics, enhancing their overall operational efficiency. In the case study of "dynamic business continuity planning during COVID-19 pandemic", the model enabled enterprises to adapt their operations, ensure business continuity, and maintain resilience amid challenging times.

To sum up, the dynamic enterprise architecture model, which integrates MAPE-K and CBR, provides a robust framework for organizations to thrive in today's unpredictable and rapidly changing business environment. By embracing adaptability and continuous learning, companies can position themselves as agile and forward-thinking entities, ready to excel in an ever-evolving world.

To further improve the model, future research may focus on enhancing machine learning techniques, validating the model's effectiveness in real-world scenarios across different industries and organizations, integrating it with emerging technologies like artificial intelligent (AI), internet of thing (IoT), and blockchain, enriching its knowledge base continuously, addressing scalability challenges, and optimizing its performance. During times of crisis, such as the COVID-19 pandemic, the proposed dynamic enterprise architecture model offers a promising solution to help organizations overcome challenges. As research and advancements in enterprise architecture continue, the proposed model has the potential to become a critical tool in ensuring the sustainability and success of organizations in an increasingly dynamic and uncertain business landscape.

REFERENCES




- [1] J. Lapalme, A. Gerber, A. Van Der Merwe, J. Zachman, M. De Vries, and K. Hinkelmann, "Exploring the future of enterprise architecture: A Zachman perspective," *Computers in Industry*, vol. 79, pp. 103–113, Jun. 2016, doi: 10.1016/j.compind.2015.06.010.
- [2] I. Ettahiri and K. Doumi, "Extended ArchiMate Metamodel with a context-awareness layer for a dynamic Enterprise architecture model," *Procedia Computer sciences, ProjMAN conference*, 2023.
- [3] G. Tamura, N. M. Villegas, H. A. Muller, L. Duchien, and L. Seinturier, "Improving context-awareness in self-adaptation using the DYNAMICO reference model," in *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, May 2013, pp. 153–16, doi: 10.1109/SEAMS.2013.6595502.
- [4] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003, doi: 10.1109/MC.2003.1160055.
- [5] K. Doumi, S. Baïna, and K. Baïna, "Modeling approach using goal modeling and enterprise architecture for business IT alignment," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6918 LNCS, Springer Berlin Heidelberg, 2011, pp. 249–261, doi: 10.1007/978-3-642-24443-8_26.
- [6] J. Saat, S. Aier, and B. Gleichauf, "Assessing the complexity of dynamics in enterprise architecture planning - lessons from chaos theory," *15th Americas Conference on Information Systems 2009, AMCIS 2009*, vol. 10, pp. 7026–7033, Jan. 2009, Accessed: Dec. 13, 2023. [Online]. Available: <https://aisel.aisnet.org/amcis2009/808>
- [7] I. Ettahiri, K. Doumi, and A. Zellou, "Towards a dynamic model of business IT alignment using enterprise architecture: a comparative study," in *Lecture Notes on Data Engineering and Communications Technologies*, vol. 152, Springer International Publishing, 2023, pp. 707–720, doi: 10.1007/978-3-031-20601-6_58.
- [8] D. B. Leake, *Chapter 1 CBR in context: The present and future*. Case-Based Reasoning: Experiences, Lessons, and Future Directions, AAAI Press/MIT Press, 1996.
- [9] I. Watson, "Case-based reasoning is a methodology not a technology," *Knowledge-Based Systems*, vol. 12, no. 5–6, pp. 303–308, Oct. 1999, doi: 10.1016/S0950-7051(99)00020-9.
- [10] J. Kolodner, *Case-based reasoning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.
- [11] A. Agnar and E. Plaza, "Case-Based reasoning: Foundational issues, methodological variations, and system approaches," *AI Communications*, vol. 7, no. 1, pp. 39–59, 1994, doi: 10.3233/AIC-1994-7104.

- [12] P. Zhang, B. Cheng, and L. Cao, "A CBR-based approach for selecting EA patterns in the context of digital transformation," in *Proceedings of the 2021 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, 2021, pp. 1–8.
- [13] L. Guo, B. Cheng, and L. Cao, "A CBR-based approach for recommending EA changes to improve the security of enterprise architectures," in *Proceedings of the 2022 IEEE International Conference on Computer Communications (INFOCOM)*, 2022, pp. 1–8.
- [14] L. Chen, B. Cheng, and L. Cao, "A CBR-based approach for managing the complexity of enterprise architectures," in *Proceedings of the 2023 IEEE International Conference on Software Architecture (ICSA)*, 2023, pp. 1–8.
- [15] J. Van Der Donckt, D. Weyns, F. Quin, J. Van Der Donckt, and S. Michiels, "Applying deep learning to reduce large adaptation spaces of self-adaptive systems with multiple types of goals," in *Proceedings - 2020 IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2020*, Jun. 2020, pp. 20–30, doi: 10.1145/3387939.3391605.
- [16] L. Fernandez Maimo, A. L. Perales Gomez, F. J. Garcia Clemente, M. Gil Perez, and G. Martinez Perez, "A self-adaptive deep learning-based system for anomaly detection in 5G networks," *IEEE Access*, vol. 6, pp. 7700–7712, 2018, doi: 10.1109/ACCESS.2018.2803446.
- [17] P. Jamshidi, J. Camara, B. Schmerl, C. Kaestner, and D. Garlan, "Machine learning meets quantitative planning: enabling self-adaptation in autonomous robots," in *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, May 2019, vol. 2019-May, pp. 39–50, doi: 10.1109/SEAMS.2019.00015.
- [18] R. D. Caldas, A. Rodrigues, E. B. Gil, G. N. Rodrigues, T. Vogel, and P. Pelliccione, "A hybrid approach combining control theory and AI for engineering self-adaptive systems," in *Proceedings - 2020 IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2020*, 2020, pp. 9–19, doi: 10.1145/3387939.3391595.
- [19] K. J. Åström and P. Eykhoff, "System identification-A survey," *Automatica*, vol. 7, no. 2, pp. 123–162, Mar. 1971, doi: 10.1016/0005-1098(71)90059-8.
- [20] A. Chiuso and G. Pillonetto, "System identification: A machine learning perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 281–304, May 2019, doi: 10.1146/annurev-control-053018-023744.
- [21] J. Sjöberg, H. Hjalmarsson, and L. Ljung, "Neural networks in system identification," *IFAC Proceedings Volumes*, vol. 27, no. 8, pp. 359–382, Jul. 1994, doi: 10.1016/s1474-6670(17)47737-8.
- [22] L. Ljung, "System identification," *The Control Handbook: Second Edition*. Wiley, pp. 57-1-57–39, May 2018, doi: 10.1007/978-3-030-42810-5_8.
- [23] L. C. L. Zhu, and B. Cheng, "A MAPE-K-based approach for managing the self-adaptability of EA models," in *Proceedings of the 2021 IEEE International Conference on Services Computing (SCC)*, 2021, pp. 1–8.
- [24] L. C. L. Liu and B. Cheng, "A MAPE-K-based approach for managing the deployment and maintenance of EA solutions in a cloud environment," in *Proceedings of the 2022 IEEE International Conference on Cloud Computing (CLOUD)*, 2022, pp. 1–8.
- [25] L. C. L. Wang and B. Cheng, "A MAPE-K-based approach for managing the evolution of EA models in a DevOps environment," in *Proceedings of the 2023 IEEE International Conference on Software Engineering (ICSE)*, 2023, pp. 1–8.
- [26] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly: Management Information Systems*, vol. 28, no. 1, pp. 75–105, 2004, doi: 10.2307/25148625.
- [27] J. Pries-Heje, R. Baskerville, and J. Venable, "Strategies for design science research evaluation," *16th European Conference on Information Systems, ECIS 2008*, vol. 87, 2008, Accessed: Dec. 13, 2023. [Online]. Available: <http://aisel.aisnet.org/ecis2008/87>
- [28] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, Dec. 2007, doi: 10.2753/MIS0742-1222240302.
- [29] I. Ettahiri, K. Doumi, and N. Falih, "A new approach for a dynamic enterprise architecture model using ontology and case-based reasoning," in *International Conference on Enterprise Information Systems, ICEIS - Proceedings*, 2022, vol. 2, pp. 553–560, doi: 10.5220/0011079400003179.
- [30] I. Ettahiri and K. Doumi, "Dynamic enterprise architecture planning using case-based reasoning and blockchain," *Procedia Computer Science*, vol. 204, pp. 714–721, 2022, doi: 10.1016/j.procs.2022.08.086.
- [31] H. Happel and S. Seedorf, "Applications of ontologies in software engineering," *Engineering*, pp. 1–14, 2006, doi: 10.1111/j.1463-1326.2004.00392.x.
- [32] S. R. Mirsalari and M. Ranjbarfard, "A model for evaluation of enterprise architecture quality," *Evaluation and Program Planning*, vol. 83, p. 101853, Dec. 2020, doi: 10.1016/j.evalprogplan.2020.101853.
- [33] L. Vinet and A. Zhedanov, *A "missing" family of classical orthogonal polynomials*, vol. 44, no. 8. Springer Berlin Heidelberg, 2011, doi: 10.1088/1751-8113/44/8/085201.
- [34] T. Lumor, A. Hirvonen, and M. Pulkkinen, "The role of enterprise architecture in building and sustaining it - Enabled organizational agility," in *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2021, vol. 2020-Janua, pp. 6057–6066, doi: 10.24251/hicss.2021.732.
- [35] G. Antunes, M. Bakhshandeh, R. Mayer, J. Borbinha, and A. Caetano, "Using ontologies for enterprise architecture integration and analysis," *Complex Systems Informatics and Modeling Quarterly*, no. 1, p. 1, Mar. 2014, doi: 10.7250/csimq.2014-1.01.
- [36] R. Schmidt and A. Kieninger, "DYNSEA - A dynamic service-oriented enterprise architecture based on S-D-logic," in *Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOC*, Sep. 2009, pp. 290–296, doi: 10.1109/EDOCW.2009.5331982.
- [37] A. D. Koffi, "Enterprise architecture dynamic alignment model," *Journal of Enterprise Architecture*, vol. 13, Oct. 2017, Accessed: Dec. 13, 2023. [Online]. Available: <https://fridaycrm.com/blog/wp-content/uploads/2021/07/enterprise-architecture-dynamic-alignment-model.pdf>
- [38] R. van de Wetering, "Dynamic enterprise architecture capabilities: Conceptualization and validation," in *Lecture Notes in Business Information Processing*, vol. 354, Springer International Publishing, 2019, pp. 221–232, doi: 10.1007/978-3-030-20482-2_18.
- [39] R. van de Wetering, "Dynamic enterprise architecture capabilities and organizational benefits: an empirical mediation study," *Twenty-Eighth European Conference on Information Systems (ECIS2020)*, no. May, pp. 1–18, May 2020.
- [40] R. van de Wetering, "Enterprise architecture resources, dynamic capabilities, and their pathways to operational value," in *International Conference on Information Systems (ICIS) 2019*. At: Munich, Germany, 2019.
- [41] N. A. Abu Bakar, S. Yaacob, S. S. Hussein, A. Nordin, and H. Sallehuddin, "Dynamic metamodel approach for government enterprise architecture model management," *Procedia Computer Science*, vol. 161, pp. 894–902, 2019, doi: 10.1016/j.procs.2019.11.197.




- [42] S. Kotusev, "Article different approaches to enterprise architecture," *Journal of Enterprise Architecture*, vol. 12, no. 4, p. 9, 2016.
- [43] R. Wagter, M. Van den Berg, J. Luijpers, and M. Van Steenberg, *Dynamic enterprise architecture how to make it work*. John Wiley & Sons, 2005
- [44] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides, "Machine learning-based predictive control of nonlinear processes. Part I: Theory," *AIChE Journal*, vol. 65, no. 11, Aug. 2019, doi: 10.1002/aic.16729.

BIOGRAPHIES OF AUTHORS



Imane Ettahiri    She received her Eng. degree in software engineering from the High National School of Computer Science and System Analysis (ENSIAS) at Mohammed V University in Rabat, Morocco in 2014. Currently, she is a Ph.D. student in the Alqualsadi Research Team at ENSIAS, Mohammed V University in Rabat. Additionally, she is a part-time professor at Mohammed V University. Her research interests include dynamic enterprise architecture, its quality, integration, and evaluation. She can be contacted at email: i.ettahiri@gmail.com.



Karim Doumi    is an associate professor at Mohammed V University in Rabat, Morocco. He received his PhD in software engineering from the Alqualsadi Research Team at the High National School of Computer Science and System Analysis (ENSIAS), Mohammed V University in Rabat, in 2013. His research interests include enterprise architecture, adaptability, agility in enterprise architecture, and business IT alignment. He can be contacted at email: k.doumi@um5r.ac.ma.