Dynamic composition components based on machine learning: architecture design and process

Younes Zouani[1], Abdelmounaim Abdali[2], Charafeddine Ait Zaouiat[3]
[1,2]Laboratory (LAMAI), Faculty of sciences and technology Cadi Ayyad University, Morocco
[3]Polydisciplinary faculty of sidi bennour, chouaib Doukkali University, Morocco

Article Info

Article history:

Received Jan 30, 2021
Revised Mar 31, 2021
Accepted Apr 11, 2021

Keywords:

Composition of components
artificial intelligence
Context-aware
e-Learning
User goal

ABSTRACT

The dynamic composition of components is an emerging concept that aims to allow a new application to be constructed based on a user's request. Three main ingredients must be used to achieve the dynamic composition of components: goal, scenario, and context-awareness. These three ingredients must be completed by artificial intelligence (AI) techniques that help process discovery and storage. This paper presents framework architecture for the dynamic composition of components that can extract expressed goals, deduce implicit ones using AI. The goal will be combined with pertinent contextual data, to compose the relevant components that meet the real requirements of the user. The core element of our proposed architecture is the composer component that (i) negotiate user goal, (ii) load the associated scenarios and choose the most suitable one based on user goal and profile, (iii) get binding information of scenario's actions, (iv) compose the loaded actions, and (v) store the new component as a tree of actions enabled by contextual or process constraint. In our e-learning proven of concept, we consider five components: composer component, reader component, formatter component, matcher component, and executor component. These five components stipulate that a course is the combination of existing/scrapped chapters that have been adapted to a user profile in terms of language, level of difficulty, and prerequisite. The founding result shows that AI is not only an element that enhances system performance in terms of timing response but a crucial ingredient that guides the dynamic composition of components.

Corresponding Author:

Younes Zouani
Department of Computer Engineering
112 Boulevard Abdelkrim Al Khattabi, Marrakesh 40000, Morocco
Email: zouani.younes@gmail.com

1. INTRODUCTION

One of the ultimate targets of software development is the optimal reuse of services, components and APIs. To achieve this goal, two proven concepts have been adopted by both academia and industry, namely component-based development engineering (CBDE) and service-oriented architecture (SOA). CBDE allows us to break down a system into components that encapsulate a set of services. Each component provides an interface that displays the services provided by this component. CBDE changes our vision of reuse, and represents a paradigm shift from reuse of a single service to a set of semantically linked services. SOA solves the problem of interoperability, allowing services to communicate regardless of the details of their implementation. However, several questions remain, such as which components are to be composed, which enhancements can be applied to the composition process that are appropriate for the user's profile, and how the explicit and implicit goals of the user can be detected. The semantic part of our approach is related to the user's goals, which cannot be clearly understood outside of the user's context. By using the concept of context awareness, the system can frame and complete the understanding of the real goals of the end user, and this can affect system requests, propositions and their formats. In order to achieve the dynamic composition of components, each goal must be associated with a set of scenarios and depending to user constraints, scenario process and contextual data; thus a particular scenario will be adopted. Artificial intelligence is a critical ingredient that interferes in the hall composition process: (i) goal negotiation; (ii) process proposal; and (iii) composition plan storage and proposal.

In total, four aspects must be considered in order to achieve dynamic composition of components: context awareness, the user's goal, goal's scenarios and artificial intelligence. In practice, SOA and CBDE work together to give a universal environment for component construction (CBDE) and component communication (SOA).

In the literature many investigations and surveys were involved about dynamic composition of components, starting with WANG *et al.* [1] who discuses the decomposition of a service to a set of Web Services and using semantic similarity we could compose. In [2] Chang *et al.* present a design of a dynamic composition handler on enterprise service bus and analyzes different types of service compositions to clarity what dynamic composition really holds in service-oriented computing. At his turn Fortuna *et al.* in [3] proposes a framework for dynamic composition of communication services that is well suited for facilitating research and prototyping on real experimental infrastructures of remotely configurable embedded devices. Elahraf *et al.* in [4] present an integrated approach that facilitates the dynamic composition of an executable response process. The proposed approach employs ontology-based reasoning to determine the default actions and resource requirements for the given incident and to identify relevant response organizations based on their jurisdictional and mutual aid agreement rules.

Several architectures and frameworks [5]-[10] have been constructed to modilise dynamic composition, Ibrahim *et al.* in [11] propose a meta middleware solution to support the dynamic composition and they propose a process of four steps: translation, generation, evaluation and building. Translation involves transforming the request into a message that is comprehensible to the system, while in the generation stage, the system attempts to generate one or more composition plans. Based on these plans, the evaluator chooses the most suitable plan based on the user context. Finally, the builder executes the selected plan and generates the associated composite.Translation involves transforming the request into a message that is comprehensible to the system, while in the generation on the other hand, recent research used case-based reasoning (CBR) efficiently in dynamic (or semi- dynamic) web service composition.CBR is one of the preferred problem-solving strategies and machine learning techniques in complex and dynamically changing situations [12]. It consists of four basic steps: (i) find similar case (Remind), (ii) adapt the solution to the specific problem (Adapt), (iii) verify that this solution works or is reasonable (Evaluate), and (iv) save this new case/solution pair for future use (Store). Case-based reasoning is a problem solving paradigm that in many respects is fundamentally different from other major AI approaches [13], [14]. In CBR, the primary knowledge source is a memory of stored cases (case base) recording specific prior episodes. The processes involved in CBR can be described by: A new problem is matched against cases in the case base and one or more similar cases are retrieved.

Briefly, if we have done a deep study of this works, we conclude that several problems are associated with these architectures/frameworks: (i) the composition of components is presented as a sequential, single-step operation that takes in the context and user goal, produces a composition plan and executes it; (ii) the context and goal are generally misused or used interchangeably; (iii) there is an absence of a clear strategy to store composition plan that enables plan proposal; (iv) there is an absence of a mechanism that can empower the reasoning capability of the system in terms of goal negotiation, process discovery and proposal.

Thus, in our turn, this paper aims to present a smart dynamic composition of components architecture based on user goal and context, involving machine learning algorithms to frame goal negotiation and to perform process proposal. Moreover, we suggest a meta-model of composition plan in order to efficiently store and propose the composition result.

2. RESEARCH METHOD

Four aspects must be considered in order to achieve dynamic composition of components: context awareness, the user's goal, the goal's scenario, artificial intelligence, and SOA/CBDE. A motivating scenario is given to understand the dynamic composition of components and how the four aspects could enhance the composition logic.

2.1. Background concepts

2.1.1. Context awareness approach

Context awareness is an important concept that is used to improve the user experience and to deliver a high-quality product that perfectly matches the user's goals. A logical definition was given by Dey *et al.* [15], who clarified the term 'context' as follows: "any information collected to define the status of an entity. An entity is a person, object or environment that is considered relevant to the interaction between an application and a user, including the user and the applications themselves". In their work, these authors illustrated the context based on what is relevant to the interaction between the user and the application [16].

The meta-modeling of context and its categorization in order to describe the inner structure of contextual data, can be very helpful in managing contextual data in a more accurate way. A lower level of contextual adaptation is the addition of fragments of code, which enable the extension of contextual adaptations by directly hard-core new ones [17]. A context awareness approach can also be used for the adoption of architecture-level techniques such as middleware or component-based architectures. It can also be implemented with proper constructions at the level of the programming language. Context-oriented programming (COP) is a new paradigm for the implementation of this type of software, and is especially applicable in the field of mobile and pervasive computing [18]. The concept of COP is used to tackle the development of contextual systems at the language level, by adding ad hoc language abstractions to handle the modeling of adaptations and their activation in a dynamic way [19].

2.1.2. Goal and scenario fragment

A goal is determined as "something that some stakeholder hopes to achieve in the future" [20]. It is presented as a clause containing a main verb and several parameters, where each parameter plays a different role with respect to the verb. Each parameter also has a semantic function, providing answers to the various different questions that can be related to this verb: who, what, when, how much and how. In order to answer the abstract question of what the goal consists of, several works have proposed a meta-model approach. A meta-model makes it possible to represent the intentions of the user and the objectives of the services. In this model, a goal is expressed by a verb, a target and one or more parameters, which can be categorized as 'direction', 'ways', 'time', 'beneficiary', 'quality', 'quantity' and 'location'. The verb and target are mandatory, while the parameters are optional. In general, any sentence can be expressed in the formalism of the target, making it possible to represent both the needs of the user and the objective that the intentional services can achieve [21].

A scenario is defined as "a possible behavior limited to a set of desired interactions between several agents" [22], and is composed of one or more actions, each of which is an interaction between one agent and another. A distinction is made between normal and exceptional scenarios: the former leads to achievement of the associated objective, while the latter fails to achieve the objective [23]. The normal scenario achieves the desired goal, while the exceptional one ends without reaching it. The actions can be categorized into two types: atomic and flux. An atomic action is an interaction between two agents that affect an object. An agent and an object may take part in several different interactions. A flow of actions is used to define the scheduling between interactions in a scenario, and is composed of several actions. The action's flows are classified into four types: sequence, competition, repetition or constrain [23].

The goal of the user can help us to determine which components must be composed in order to satisfy this goal. The main objective of understanding the expressed goal of the user is to transform this goal into a concrete composition plan, and then to produce a workflow for composing components to match the final goal. However, a full understanding of the user's aspirations cannot be achieved without considering context awareness, since two users with the same expressed objective maybe have differently depending on the parameters of the context, such as non-expressed information, profession, age, gender, culture, knowledge and preferences. Context awareness is therefore a critical paradigm that complements and can redirect the user's goal [24].

2.2. Soa and cbde

SOA is a set of standardized functions that allow developers to achieve their aims using the capabilities they have, regardless of the environment in which they are located, and these capabilities can be organized or combined for maximum business benefit [25]. Through the use of an SOA, a service can be described and discovered, and can communicate with other services, regardless of heterogeneities in implementation. In many ways, the terminology used in relation to services is much the same as that used to define component-based development; however, certain specific terms are used to define elements within web services. The service provider publishes the web service to a discovery agency. A potential service consumer searches for a service from the discovery agency, acquires the URL of the required service, obtains the WSDL file, builds the client, and uses the service provided [26].

CBDE is another paradigm that uses SOA. By composing existing or customized components, a software system can be assembled as rapidly and cost-effectively as an automobile is assembled by composing machine parts [27], while SOA focuses on transforming the process implementation into a technological-independent solution. From the perspective of the composition of components, SOA/SCBDE provides a universal platform for creating, composing and deploying components and exposing components service using different SOA implementations.

2.3. Layered architecture for dynamic composition of component

2.3.1. High level architecture design

Our architecture is based on five layers as shown in Figure 1, a graphical user interface (GUI), a context manager (CM), a goal manager (GM), a scenario-context manager (SCM), and a composition manager (CoM).

a) GUI layer: The interaction end-user/system is a critical part of the composition process (the translation aspect), as it helps the end user to understand the ambiguities of the system and to adapt different requests to the user context. To express the needs of the end user, the literature distinguishes between internal and external specification languages. In our proposed scheme, we express the user goal using a GUI composed of four parts: a business GUI, a contextual GUI, a component registration GUI and a programming GUI. The business GUI is responsible for grouping components by domain area, and enables a component search based on the goal keyword. The main task of the contextual GUI is to display the user’s context attributes, i.e. the contextual attributes that are required by a specific component and the contextual values that will be sent during the composition process. The component registration GUI is used to improve our system by registering/altering components, their interfaces, the scenarios, and their associated contexts. The programming GUI is a business helper that performs basic programming actions, such as the repetition of a business action (a simplified graphical loop) and testing the result of the execution of a process (graphical if).

b) CM layer: While the GUI layer handles the contextual parameters to be input to the composition process, the mission of the CM layer is to generate these parameters by collecting contextual data and analyzing them in respect to the user back-ground and culture [28]. Contextual data exist everywhere, for example in the user profile, external sensors, historical events and the compositions of similar users. There are two types of contextual data: CRUD contextual data and deduced contextual data. The former represents data that have been directly received from the environment, such as the user profile, hardware and weather, while the latter requires more processing and system intelligence to extract them, and can be used to enhance the composition process.

c) GM layer: This layer involves composing the sub-goals in order to meet the final goal of the end user. The overall goal may be superficial, incomplete or even wrong, and the GM layer intervenes to guide the end user based on (i) user similarities, and (ii) a goal or component that has been registered by a domain expert, in order to help other users to conceive and correct their initial goals.

d) SCM layer: The goal must be adapted to the specificity of each user. The SCM layer takes the user goal as input and looks up the associated scenario in order to enhance the scenario process by contextual parameters.

e) CoM layer: After constructing the composed goal and enhancing the global scenario using contextual parameters, the system must compose the associated software component in order to meet the user constraint. The component registration step assumes that an expert user associate component facade, goal and context parameters. The task of this layer is to construct and store the new component.

Our architecture is based on constructing/negotiating the goal with a user-friendly demarche (GUI), which enables a simple, white-box intervention by the end user in the composition process. The constructed goal is also associated with a global scenario that can be decorated and enhanced by contextual parameters. Finally, the new component that performs the requested goal is constructed and stored.

2.3.2. Low level architecture components and process

A domain expert is a user with both domain and technical knowledge which allows him to perform the environment preparation by (i) setting goals for a specific domain, (ii) building scenario process and defining associated bag of words to facilitate its discovery, (iii) empowering each scenario with contextual data to take environment change into account, and (iv) linking each scenario process with a set of component facades that represent the concrete implementation of the process. From an end-user point of view, the composition process is made up of four steps: (i) each scenario/goal is associated with a bag of words that facilitate it search, (ii) once the scenario/goal is founded, either associated scenarios are proposed using AI techniques or a predefined goal is selected, (iii) the user request an abstract composition plan based on selected scenarios therefore the system must perform a concrete composition by selecting the most suitable façade's services using reinforcement learning (iv) finally, the system load components associated with the selected scenarios, perform dynamic composition of components and register the composition as a decision tree. This decision tree will be used later in a random forest algorithm in order to propose a goal in step (iii) of the process.

The inner structure of our proposed architecture is based on four modules as shown in Figure 2, (i) Component Module to describe the component structure and semantic, (ii) Goal Module which is an action performed by an actor, (iii) Scenario Module that holds a bag of words to facilitate it discovery and finally (iv) Composition Module.

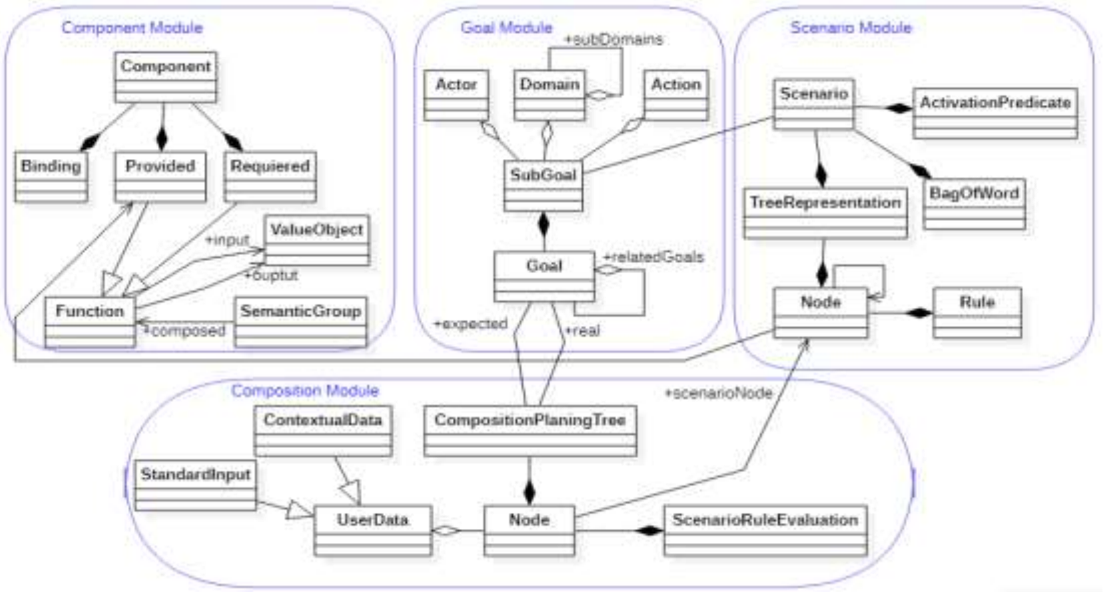

Figure 1. The meta-model of the composition of components framework

a) Component module

The Component module is used to present a component inner structure. In fact, a component is defined by three main elements: provided interface, required interface and binding information. Provided interface sum-up services that are provided by the component, whereas the required interface holds reference of external services consumed by the component. To simplify the complexity of input and output data of the provided and required interfaces, a value object is introduced as a set of primitive attributes to encapsulate the complexity of inputs/outputs, thus a process of mapping of value object to input/output and vice versa must be envisaged. In order to accelerate its discovery, provided interface element are grouped by semantic.

b) Goal module

The goal module defines a goal as a composition of sub-goals; each sub goal is an action performed by an actor and semantically related to a specific domain. Each goal is semantically related to other goals: travelling to another county is semantically related to booking a room in a hotel or renting a car. To facilitate the discovery of a goal based on its domain, a domain is decomposed on sub-domains.

c) Scenario module

Since goal is an abstract concept, scenario is a more pragmatic concept that has been introduced to dynamically compose component. In our meta-model we consider that a scenario is a bag of words, an activation predicate and a tree representation. The tree representation is a binary tree where each node holds a validation rule that decides the next node to be chosen. On the other hand, the bag of words is a set of words with the associated probabilities that represent the representativeness of a word in the scenario. For example, the bag [(object,0.9) ; (constructor,0.7) ; (inheretence,0.6) ; (encapsulation,0.5) ; (class,0.8)] represent the most accurate word that reflect the scenario of learning an object oriented course. The activation predicate is the validation condition to lunch the scenario. In order to study an object oriented course, the validation predicate could be having algorithmic prerequisite.

d) Composition module

Composition module conceives the composition planning as a tree that measure the distance between the expected goal and the real goal. Each node holds three main ingredients: user data, scenario node and the scenario rule evaluation. In fact, two categories of user data must be distinguished: standard input and contextual data. Standard inputs are mandatory data that must be provided by the end user in order execute a specific scenario, while contextual data are optional information that frame and guide the scenario execution.

3. RESULTS AND DISCUSSION

3.1. Application of the meta model

In the context of e-learning, each course is represented by a component which exposes the services providing its chapters, the associated translations and the prerequisites required. The scenario associated with the objective "studying a course" is as follows: (i) the collection of knowledge, (ii) the processing of knowledge, (iii) the construction of knowledge and (iv) the derivation of knowledge.

a) Collection: when the user looks for a specific course, our system consult the local course database and online courses in order to select the most suitable chapters based on user domain knowledge, and prerequisite.
b) Processing: the first element of this step is formatting which consist on unifying the format of the collected chapters (converting video/audio chapters to doc/pdf file for example). Once data have been formatted, collected data must be adapted to user preference such as translation from the original language to user preferred language and a practical oriented course or a classic one.
c) Construction: Once collected chapters have been formatted and adapted, the system registers the constructed course as single component, thus this component could be consulted by user with the same requirements and profile.
d) Derivation: Based on a registered course, the system must be able to derive synthetic knowledge from existing one. In order to facilitate and control the process of knowledge transmission, the system must be able to sum up and extract quiz or QCM from the constructed course.
e) Composition: while the result of previous steps is a consistent course component with the derived knowledge, this step is interested in building an optimal combination of chapter resulting from several components (course) based on the client's request and its context.

3.2. E-learning application results and discussion

To evaluate our proposed architecture in the context of e-learning application, we have built five APIs: reader component API, formatter component API, matcher component, executor component API and the dynamic composer as shown in Figure 2. The reader component API is interested on scrapping online courses based their summary page and unify theirs formats since courses could be html pages and pdf files. On the other hand, the executor component holds core domain business to be executed. In e-learning scenario, it performs course composition based on scrapped courses and contextual parameters (language preference, level of difficulty) by composing the most appropriate chapters. Two helpers could be used in the composition process: matcher component API and formatter component API.

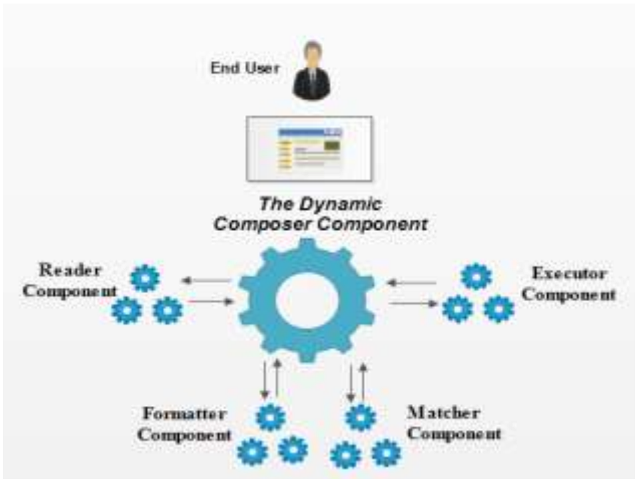

Figure 2. The dynamic composer for course composition

While the formatter component API allows us to adapt the chapters of a course to the user's context by converting from one language to another (from French to English), the matcher component API ensures that two chapters are equivalent using their bag of words, so that we can select the most relevant one for a well-defined user.

In our test, we assume that the goal and the associated scenario are already predefined. The goal is getting a course by combining the most suitable chapters of different courses, whereas the associated process is: (i) looking for at least three courses that meets end-use key word in the scrapper data base or scrapping it if it doesn't exist, (ii) translate and format the scrapped courses, and (iii) select and compose the course based on scrapped chapters and user context.

To test the performance of the dynamic composition composer we perform six requests of distinguished by the level of difficulty and the language of the end-user:

a) Request 1: easy, English JAVA course.
b) Request 2: medium, French JAVA course.
c) Request 3: difficult, English JAVA course.
d) Request 4: easy, French JAVA course.
e) Request 5: medium, Arabic JAVA course.
f) Request 6: difficult, Arabic JAVA course.

The dynamic composer component is the core component of the architecture since it takes the composition plan and synchronies between different components to achieve the selected scenario. For the composer component, achieving dynamic composition is a process that's applied to the triplet (URL, Request j, OPERATIONS (Request 1….j-1)). In fact, URL represent a unique identifier to access to a component's action, while the Request j is either a client request that holds business and contextual data, or the result of logical/arithmetical operations applied to previous requests.

The performance of execution of the dynamic composer is depicted in the Figure 3.Two categories of request could be distinguished: Request 1, 2 and 3; and Request 4, 5 and 6. In fact, the first category performs scraping, translation and composition of easy, medium and difficult courses, while the second one translates the scrapped courses from its original language to the end-user one which justify the low cost of the second category. For instance, the first request seeks for English and easy JAVA course, the result offered by the scrapper API is two English courses and 3 French ones. Therefore, the Translator API will translate the 3 French courses and compose it to get a composed course made up of the most ratted chapters of the five courses. The fourth request looks for French and easy JAVA course so the composer will use the result of the first request and the Translator will translate the three English courses to French.

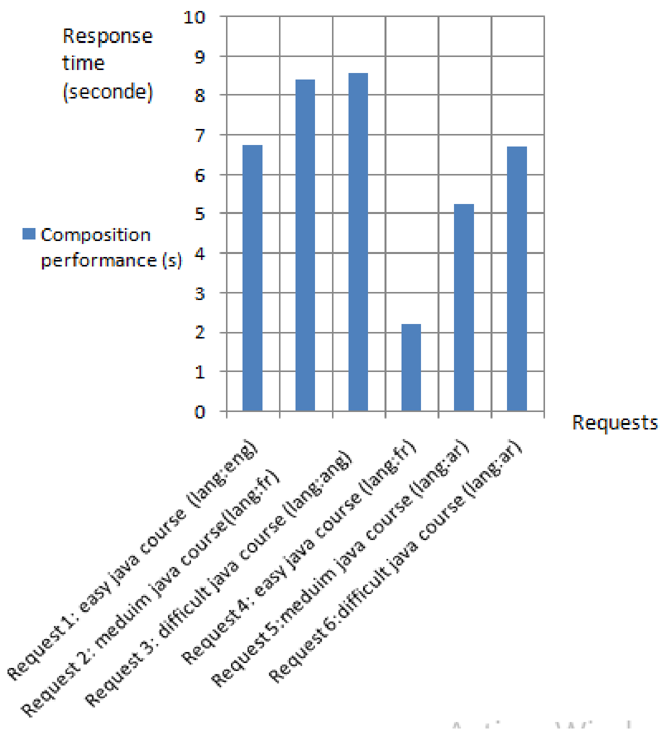

Figure 3. The dynamic component composer performance applied to course composition

4. CONCLUSIONS

In this paper, we present an approach that can combine context, the user goal and the scenario in the composition process. This approach is based on a layered architecture composed of scenario discovery, context collection, and scenario-context tree construction. From a user perspective, the composition process start with the discovery of basic scenarios using a bag of words then, the end-user perform a dynamic composition of founded scenario using mandatory parameters and contextual data and finally, the composition result is registered as a new scenario. Our dynamic composer performs composition in four steps: (i) reading core and contextual data, (ii) Formatting data to fit to user context, (iii) matching component to be composed and (iv) composition execution based on restful data (URI and data) of each component. For the e-learning scenario, we have proposed five main steps: (i) collection, (ii) processing, (iii) construction, (iv) derivation, and (v) composition. The process trigger is the end-user key word related to a specific domain, then the knowledge collector assembles relevant chapters from different courses, after that, collected chapters are processed in order to be formatted and adapted to the user context, so a new course could be constructed. The last step is about deriving summaries, multiple choice questions and exams to control and facilitate the learner's understanding.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge support from Prof. Abdelmounaim ABDALI, Prof. Charafeddine Ait ZAOUIAT and the members of LAMAI (Laboratory of Mathematics Applied and Informatics) within the Faculty of Science and Technologies, Cadi Ayyad University, Marrakesh.

REFERENCES

[1] Wang, Y. “Research on web service composition algorithm using description logic,” *Telk. Indones. J. Electr. Eng,* vol. 12, pp. 852-858, 2014, doi: 10.11591/telkomnika.v12i1.3536.
[2] Chang, Soo Ho, *et al*., "Design of a dynamic composition handler for esb-based services." *IEEE International Conference on e-Business Engineering (ICEBE'07)*. IEEE, 2007, doi: 10.1109/ICEBE.2007.63.
[3] Fortuna, Carolina Et Mohorcic, Mihael, “A framework for dynamic composition of communication services,” *ACM Transactions on Sensor Networks (TOSN)*, 2014, doi: 10.1145/2678216.
[4] Elahraf, Abeer, Afzal, Ayesha, Akhtar, Ahmed, *et al*., “A Framework for Dynamic Composition and Management of Emergency Response Processes,” *IEEE Transactions on Services Computing*, 2020, doi: 10.1109/TSC.2020.3030211.
[5] S. Kalasapur, M. Kumar, and B. A. Shirazi, “Dynamic service composition in pervasive computing,” *IEEE Trans. Parallel Distrib. Syst.*, 2007, doi: 10.1109/TPDS.2007.1039.
[6] T. G. Stavropoulos, D. Vrakas, and I. Vlahavas, “A survey of service composition in ambient intelligence environments,” *Artificial Intelligence Review*, 2013, doi: 10.1007/s10462-011-9283-1.
[7] K. Fujii and T. Suda, “Semantics-based context-aware dynamic service composition,” *ACM Trans. Auton. Adapt. Syst.*, 2009, doi: 10.1145/1516533.1516536.
[8] Ourania Hatzi, Dimitris Vrakas, Mara Nikolaidou, Nick Bassiliades, Dimosthenis Anagnostopoulos, and Ioannis Vlahavas, “An integrated approach to automated semantic web service composition through planning,” *IEEE Transactions on Services Computing*, vol. 5, no. 3, pp. 319-332, 2012, doi: 10.1109/TSC.2011.20.
[9] M. Kellar, H. Stern, C. Watters, and M. Shepherd, “An information architecture to support dynamic composition of interactive lessons and reuse of learning objects,” in *Proceedings of the Hawaii International Conference on System Sciences*, 2004, doi: 10.1109/HICSS.2004.1265048.
[10] Veeresh, P., R. Praveen Sam, and C. Shoba Bin, "Reliable fault tolerance system for service composition in mobile Ad Hoc network, “*International Journal of Electrical & Computer Engineering (2088-8708),* vol. 9, no. 4, 2019, doi: 10.11591/ijece.v9i4.pp2523-2533.
[11] Ibrahim, Noha Et Mouel, Frederic Le, “A survey on service composition middleware in pervasive environments,” arXiv preprint arXiv:0909.2183, 2009.
[12] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, DanielWeld, and David Wilkins, “PDDL-the planning domain definition language,” 1998.
[13] S. Ben Mokhtar, D. Fournier, N. Georgantas, and V. Issarny, “Context-aware service composition in pervasive computing environments,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2006, doi: 10.1007/11751113_10.
[14] Purohit L., Chouhan S. S., Jain A, “ Dynamic Web Service Composition Using AI Planning Technique: Case Study on Blackbox Planner,” In: Shukla R., Agrawal J., Sharma S., Chaudhari N., Shukla K. (eds) *Social Networking and Computational Intelligence*. Lecture Notes in Networks and Systems, vol 100. Springer, Singapore, 2020, doi: 10.1007/978-981-15-2071-6_15.
[15] K. Dey, “Providing Architectural Support for Building Context-Aware Applications,” PhD thesis, College of Computing, Georgia Institute of Technology, 2000.
[16] W. Liu, X. Li, and D. Huang, “A survey on context awareness,” in *2011 International Conference on Computer Science and Service System, CSSS 2011-Proceedings*, 2011, doi: 10.1109/CSSS.2011.5972040.

[17] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung, "Ontology based context modeling and reasoning using OWL," in *Proceedings - Second IEEE Annual Conference on Pervasive Computing and Communications, Workshops, PerCom*, 2004, doi: 10.1109/PERCOMW.2004.1276898.
[18] Belhadi, Siham and Rachid Merzougui. "Design and implementation of a context-aware health service platform (CAHS)." *International Journal of Electrical & Computer Engineering* (2088-8708), vol. 9,no. 6, 2019, doi: 10.11591/ijece.v9i6.pp4993-5005.
[19] G. Salvaneschi, C. Ghezzi, and M. Pradella, "Context-oriented programming: A software engineering perspective," *J. Syst. Softw.*, 2012, doi: 10.1016/j.jss.2012.03.024.
[20] Ahmad, Outfarouin and Abdali AbdelmounaÎm, "On a New Modeling Process of the Decision-Makers' Needs," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 17, no. 2, p. 1, 2017.
[21] Prat, Nicolas, "Réutilisation de la trace par apprentissage dans un environnement pour l'ingénierie des processus," Diss. Paris 9, 1999.
[22] Ben Achour, Camille, "Extraction des besoins par analyse de scénarios textuels," Diss. Paris 6, 1999.
[23] P. R. Lumertz, L. Ribeiro, and L. M. Duarte, "User interfaces metamodel based on graphs," *J. Vis. Lang. Comput.*, 2016, doi: 10.1016/j.jvlc.2015.10.026.
[24] K. Verbert *et al.*, "Context-aware recommender systems for learning: A survey and future challenges," *IEEE Transactions on Learning Technologies*, 2012, doi: 10.1109/TLT.2012.11.
[25] Parnianifard, A. S. Azfanizam, M. K. A. Ariffin, and M. I. S. Ismail, "An overview on robust design hybrid metamodeling: Advanced methodology in process optimization under uncertainty," *Int. J. Ind. Eng. Comput.*, 2018.
[26] Emmanouilidis, R. A. Koutsiamanis, and A. Tasidou, "Mobile guides: Taxonomy of architectures, context awareness, technologies and applications," *Journal of Network and Computer Applications*. 2013, doi: 10.1016/j.jnca.2012.04.007.
[27] Ameller, X. Burgués, O. Collell, D. Costal, X. Franch, and M. P. Papazoglou, "Development of service-oriented architectures using model-driven development: A mapping study," *Information and Software Technology*, 2015, doi: 10.1016/j.infsof.2015.02.006.
[28] Setiawan, Alexander, Andreas Handojo, Rendra Hadi, "Indonesian Culture Learning Application Based on Android," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 1, pp. 526-535, 2017.