ISSN 2278-3091
Volume 10, No.1, January - February 2021
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse601012021.pdf
https://doi.org/10.30534/ijatcse/2021/601012021

Towards a New Framework Architecture for a Dynamic Composition of Components Using Context Awareness and User Goal

Younes Zouani[1], Abdelmounaim Abdali[1], Ahemed Outfarouin[2]
[1]Cadi Ayyad University, Faculty of sciences and technology, Laboratory (LAMAI), Marrakesh, Morocco, zouani.younes@gmail.com
[2]Ibn Zohr University, National School of Business and Management, Research Laboratory on the Saharan space (LARES), Dakhla, ah.outfarouin@gmail.com

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. This is achieved by dynamically composing and assembling disturbed components with home ones. This paper presents a framework architecture for the dynamic composition of components that can extract pertinent contextual data and combine them with explicit/implicit intent, in order to compose the relevant components to meet the real requirements of the user. The proposed architecture includes a user feedback system that is appropriate for the use context in terms of the user profile and technical/domain knowledge. Our platform can consult the end user in order to resolve eventual composition ambiguities. The dynamic aspect of our proposition involves (i) the detection of environmental changes in response to dynamic triggers; (ii) interactive adaptation to internal changes and external stimuli; (iii) determination of the real intent of the end user; and (iv) dynamic generation of different composition plans and selection of the most appropriate option, based on context data and user intent.

Key words: Dynamic Composition of Components, Context-Aware, User Goal, SOA, CBDE.

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 specific context. By using the concept of context awareness, the system can generate an 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 decomposed into sub-goals that could meet the process proposed by different components' interfaces; the overall process must then be composed to reach the final goal.

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

2. MOTIVATING SCENARIO

Alice is an IT student who wants to learn Java, so she consults our platform and types in the keyword Java. First, our system collects contextual data about Alice, such as the languages that she has already mastered, her age, her level of programming knowledge, her current location and her social position (can Alice afford an online course, and if so, how much can she pay?). Next, the system tries to understand Alice's true intent: is she looking for a course or online training, or is she just asking about latest updates and news? To determine this, the system asks her more questions to find out her goals. Finally, when our platform understands Alice's intent, which is to take part in a Java course, the system (i) collects different books and tutorials; (ii) translates them based on Alice's language preference; (iii) extracts those

parts that match her level of ability; and (iv) automatically generates evaluations (QCM, exams) to help to test Alice's understanding of the course. In order to implement the last of these processes, four main components must be composed: the acquirement component (AC), the translation component (TC), the profiling component (PC) and the evaluation component (EC). The AC aims to collect and store various courses that are related to the keyword Java (books, tutorials, videos etc.) and to sort them based on the degree of relevance. The second component (TC) translates non-textual courses to textual ones and from this text-based corpus into another language. The PC adapts the Java course to Alice's context: her age, level of knowledge, language, and the hours that she can spend on the course. This component can create a new course by combining suitable parts from different courses, depending on Alice's request. The last component uses text mining to generate a quiz and QCM in order to evaluate Alice's understanding of the course generated by the system.

3. AN OVERVIEW ON DYNAMIC COMPOSITION OF COMPONENTS

Three aspects must be considered in order to achieve dynamic composition of components: context awareness, the user's goal, the goal's scenario, and SOA/CBDE.

3.1 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.

Although several researchers have tried to formalize the definition of context awareness, a universally accepted definition has not yet been developed in the scientific community. However, certain types have been identified that are considered relevant and important.

The concept of context awareness was introduced in 1994 by Schilit et al. [1], who defined it as: "the ability of a mobile user's applications to discover and react to changes in the environment they are situated in". In this definition, Schilit presented the context in terms of three important aspects: where the user is, who they are with and what resources are nearby.

For Almutairi et al. [2], context awareness involves a "system that uses context in an effort to provide the appropriate information or service to the user when the appropriate and meaningful information depends on the requirement and need of the user".

A logical definition was given by Dey et al. [3], 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 [4].

In addition, the term 'context' may refer to a combination of different types of context, with specific values, in order to reflect a user's situation. To control the different focus types and the values entered by the focus system, a focus template is used to specify the links and the storage structure used for the different types of focus and values [5].

Several approaches to dealing with context awareness have been elaborated in the literature, in terms of contextual retrieval and handling (analyzing and adapting the core service to contextual data) [6], for example:

(1) Middleware solutions and dedicated service platforms
(2) The use of ontology
(3) Rule-based reasoning
(4) Programming/language extensions at the source code level
(5) Model-driven approaches
(6) Message interception

The aim of these solutions is to encapsulate contextual adaptation into a distinct logical unit using the semantic web (2) or a rule engine (3).

The meta-modeling of context and its categorization in order to describe the inner structure of contextual data (1) 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 ((4), (5) and (6)), which enable the extension of contextual adaptations by directly hard-core new ones [7].

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 [8]. 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. 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 [9].

In contrast, the core domain context is a semantic adaptation and enhancement that is specific to a particular application, such as highlighting a clothes item as affordable and interesting during e-shopping, based on the user's budget and style. The general context involves the static and application-independent aspects of the context, while the core domain relates to the dynamic and application-dependent aspects.

3.2 Goal And Scenario Fragment

A goal is determined as "something that some stakeholder hopes to achieve in the future" [10]. 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, how, etc. In order to answer the abstract question of what the goal consists of, several works have proposed a meta-model approach.

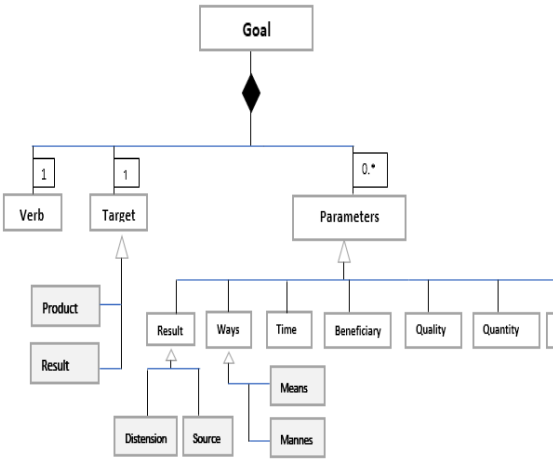

Figure 1: Meta-model of a goal.

A meta-model makes it possible to represent the intentions of the user and the objectives of the services (Fig. 1). 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 [11].

A scenario is defined as "a possible behavior limited to a set of desired interactions between several agents", and is composed of one or more actions, each of which is an interaction between one agent and another. The particular combination of actions in a scenario describes a unique pathway scenario.

Fig. 2 shows a normal scenario that achieves the desired goal, while an exceptional scenario ends without reaching the goal. 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 constraint [13].

A goal and a scenario are two complementary concepts that can be used together to compose components. The interface of each component is associated with a goal that describes the human perspective of that component, whereas the scenario associated with the goal clarifies the inner structure of the service, enabling us to understand how the goal will be achieved in practice, from a workflow perspective. A scenario workflow can be used to extend and adapt the interface of the associated component, and hence to achieve the dynamic composition of components [14].

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 may behave differently depending on the parameters of the context, such as non-expressed information, profession, age, gender, culture, knowledge, preferences, etc. Context awareness is therefore a critical paradigm complements and can redirect the user's goal [15].

Business specificities involve providing a set of semantic rules for a specific domain, in order to help the system to identify, frame and correct the user's initial goal. In general, the initial rules must be developed by a domain expert, but these rules can then evolve based on law changes, improvements to administration and work flow etc. This step corresponds to the platform learning stage.

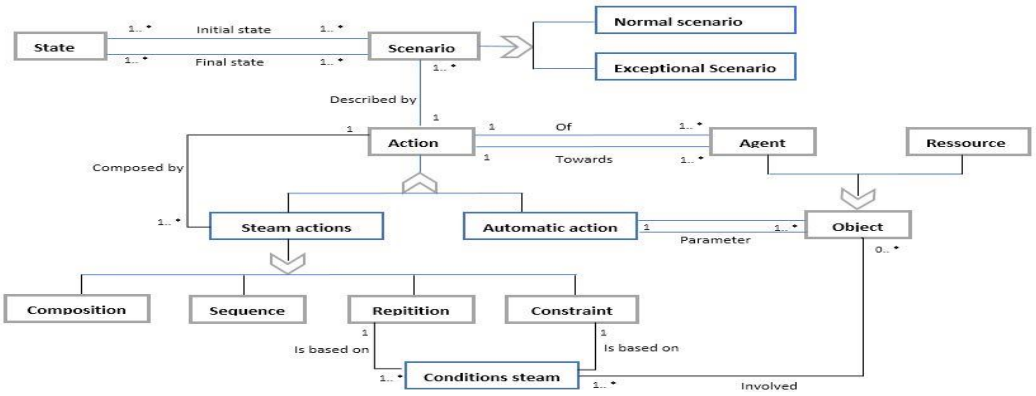

Figure 2. Meta model of a Scenario

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 [12].

3.3 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 [16].

In other words, an SOA is an emerging architectural style for developing and integrating enterprise applications. It is an organizational and technical framework that can enable an enterprise to deliver self-describing and platform-independent business functionality". 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.
Fig. 3 illustrates how public web services work. 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 [17].
CBDE is an interesting paradigm that is used to construct a new software by composing and assembling existing components.

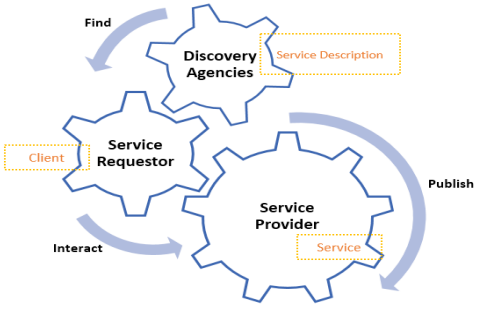

Figure 3: Principle of operation of public web services

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 [18], while SOA focuses on transforming the process implementation into a technological-independent solution. A service component architecture (SCA) [17] is a set of specifications for building distributed applications by combining SOA and CBDE

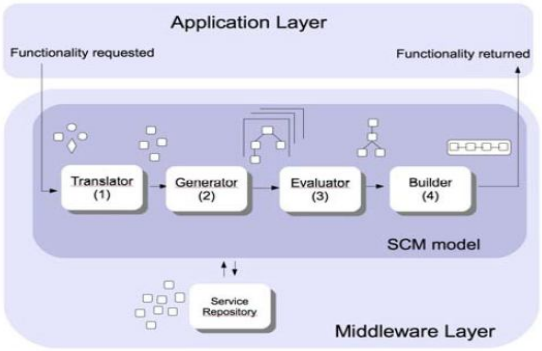

Figure 4. The four steps of Composition process [24]

with the aim of achieving technological independence and domination. SCA is designed to be independent of the programming language, binding details, communication protocols, and even the data source. From the perspective of the composition of components, SCA provides a universal platform for creating, composing and deploying components and exposing components service using different SOA implementations. Our proposition is based on the component model, in which each composition step is achieved by a composite.

Several architectures and frameworks [19],[20],[21],[22] have been proposed to deal with the dynamic composition of components and/or services, where the associated process stages are translation, generation, evaluation and building.(Fig. 4)
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.
Elahraf et al in [23] 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.
There are several problems associated with these architectures/frameworks, for example: (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 powerful concept to describe the operational structure of the abstract goal; (iv) this approach involves a passive vision of the user's role in the composition process, since he or she simply provides or expresses the composition plan; and (v) there is an absence of a mechanism that can empower the reasoning capability of the system in terms of composing components .

4 A LAYERED ARCHITECTURE FOR DYNAMIC COMPOSITION OF COMPONENT

4.1 High level Architecture Design

In this section, we will present our work in two parts. In the first, we introduce the different layers in our scheme, while in the second we explain the inner structure of the proposed system by describing the composites of the architecture.

Our architecture is based on five layers (Fig. 5): a graphical user interface (GUI), a context manager (CM), a goal manager (GM), a scenario-context manager (SCM), and a composition manager (CoM).

- 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), testing the result of the execution of a process (simplified graphical if), etc.

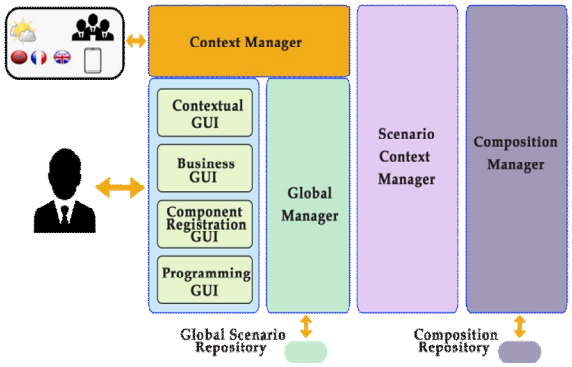

Figure 5: High level architecture design.

- 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. Contextual data exist everywhere, for example in the user profile, external sensors, historical events, the compositions of similar users, etc. 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, weather, etc., while the latter requires more processing and system intelligence to extract them, and can be used to enhance the composition process.
- 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.
- 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.
- 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, stored and deployed.

4.2 Low Level Architecture Components and Process

In order to clarify our general architecture, we explain the inner structure of each layer (Fig. 6).

A. GUI Composite

The GUI composite is formed of three components that help the user to express his or her intent and help the system to clearly understand this intent. Three kinds of user interfaces are used in this component: a standard input/output interface, a contextual user interface, and a business process interface.

- Standard input/output interface: This helps the user to express standard input/output items, such as labels, buttons and forms, and also basic programming statements such as conditions, loops, etc.
- Contextual user interface: This dynamic interface depends largely on the user profile, hardware characteristics, current environment, and various contextual changes. It aims to adapt the graphical component to the user context, and especially the user's knowledge.
- Business process interface: This includes two categories of services: generic and domain-specific services. The former are transversal services that can be used by any kind of application, such as determination of location, money conversion, language translation, etc., while the latter include component interfaces for a specific domain, predefined shared web services and the possibility of combining services using possible adaptors.

The GUI composite gives a new contextual configuration to the contextual data handler composite and helps the user intent composite to look for user intent, in order to elaborate a composition plan using the CM composite.

B. *Contextual Data Handler Composite*

The contextual data handler (CDH) composite has three components: the contextual CRUD data collector (CCDC), the contextual semantic data collector (CSDC), and the context data analyzer (CDA).

- CCDC: CRUD data are information gathered directly from sensors or the application itself (localization, user profile etc.)
- CSDC: Some intuitive information must be extracted in order to be injected in the context. This information is either the result of an analysis of CRUD data or information included in the composition process.
- CDA: This component analyzes information to obtain useful context parameters. The CDA transforms CRUD data collected by the CCDC into semantic information, which is critical in gaining an understanding of the user's objective and circumstances.

Contextual data is essential in order to determine the real intent of the user.

C. *User Intent Composite*

The user intent (UI) composite collects the CRUD and implicit objectives in order to extract atomic objectives. These are directly associated with atomic services, and thus the detection of atomic objectives and their relationships is the first step in establishing a service composition.

- CRUD objective detector: Collecting the objectives explicitly expressed by the user is the first step in understanding these objectives. The main sources of user objectives are a standard input/output interface (forms, basic programming statements etc.) and the component that has been selected to be composed in the business process interface component.
- Implicit objective generator: The expressed objectives, combined with contextual data and historical composition, form the raw materials from which the system deduces the non-expressed objectives. Using context information, the system can construct a group of similar users and use historical composition requests to resolve non-expressed objectives and propose more appropriate alternatives.
- Objective decomposer: The user's objective must be decomposed into atomic ones, in order to create the system atomic service.

The atomic service is enhanced by contextual items in order to complete and redirect each atomic objective.

The deduction of implicit intent from similar users and historical composition must be supported by the use of a predefined composition template of certain categories of users (based on their context) in order to serve the users and guide those with no clear intent in mind.

D. *Intent-Context Tree Composite*

An intent-context tree (ICT) composite is responsible for intent-context association. Context is a critical ingredient that can help in delimiting, enhancing and clarifying the user's intent. The intent-service association must therefore be preceded by context-intent association in order to enhance atomic service and atomic intent matching. The ICT composite has three main components:

- Context objective mixer: The UI composite constructs atomic objectives, whereas the CDH composite collects implicit and explicit context data; these two types of information must be associated, so that every atomic objective is matched with the linked context item.
- Ambiguity: Associating user intent with context items is not an easy task that can be performed without ambiguities. The use of historical association and eventual template can be helpful in this process. There are three possible cases: no association is possible; exactly one association is

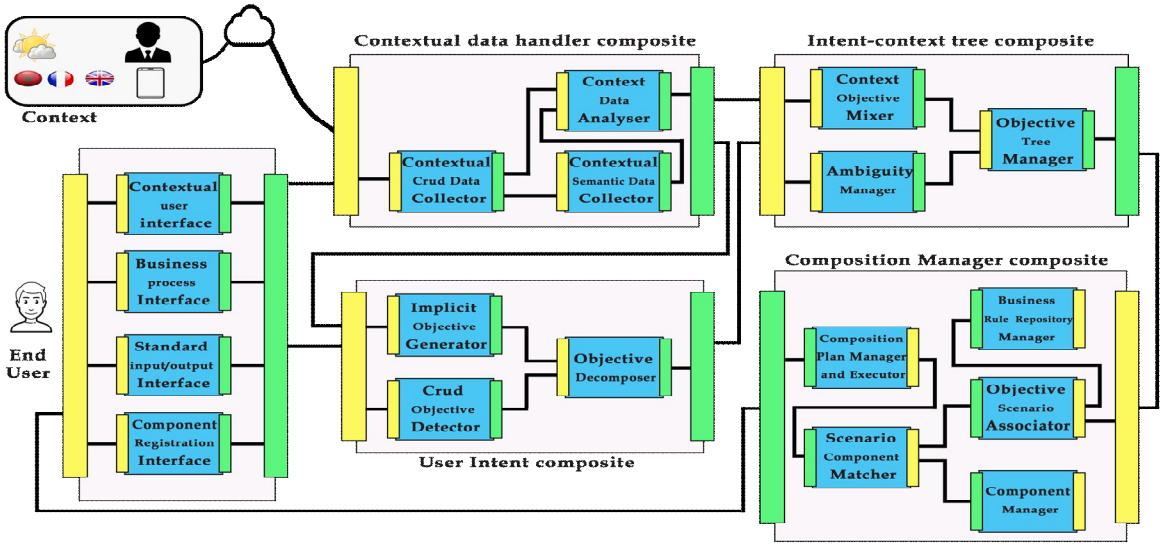

Figure 6: Composites of the proposed architecture.

possible; and more than one association is possible. The first and third cases give rise to an ambiguity problem, and user input into the association process is necessary to resolve this.

- Objective tree manager: The first step in constructing this tree is to determine composite and atomic intents, which help us to build the frame of the tree. The second step is to associate the composite and atomic intent of a specified request (the composite and atomic intent instances) with the frame constructed in the first step. Each intent instance is associated with either a contextual item or an abstract context, in order to complete and enhance the intent.

This association improves the system by giving the atomic service more input parameters in order to respond more efficiently to the user's request. The aim of organizing the intent and context data into tree form is mainly to construct a historical database that can respond and propose compositions based on context and intent similarities.

E. CoM Composite

The CoM is the backbone of our architecture, since it (i) maps the objective to a scenario based on the business rule repository and associated scenarios; (ii) maps the scenario to a suitable component in order to meet the user objectives; and (iii) chooses the most appropriate composition plan based on the suitability of the provided services and quality metrics.

- Business rule manager component: This contains the core domain business rules, grouped by domain and subdomain. Each business rule reflects a business process, and reflects a set of scenarios depending on the user's parameters. Business rules are defined, maintained and enhanced by a domain expert.

- Objective scenario associator: The association of the user objective with a predefined scenario is one of the most important steps in the composition process, since it enables a transformation from a user-dependent parameter (objective) to an element that is comprehensible to the system, in the form of a scenario. This work is performed based on two inputs: a predefined scenario for a specific business rule, and the user objective.

- Component manager: This component handles the inner and outer structure of a component. The inner structure is related to the classes of component and their interactions, whereas the outer structure relates to the configuration parameters, consumed and provided interfaces. Each component must be registered by an expert user so that it can be mapped to a specific business rule scenario.

- Scenario component matcher: Dynamic matching of a component to a specific scenario is a pivotal step in the composition process. The component and its associated services is a machine-friendly concept, whereas a scenario is a user-friendly concept; hence, combining these two elements means that a robust bridge can be created between the machine and the user. This mapping is performed based on the extracted scenario (objective scenario associator component) and the most suitable component service (component manager).

Composition plan manager and executor: When the components that match the defined scenario have been determined, the system generates a list of possible composition plans and chooses the most suitable based on the relevance of the constructed composite and the QoS parameters.

5. CONCLUSION

In this paper, we present an approach that can combine context with the user goal in the composition process. This approach is based on a layered architecture composed of context triggering, context collection, composition, and deployment processes. When the context listener composite detects changes in the environment or in the internal system itself, the contextual data handler composite gathers contextual information about the new situation.
This information must be analyzed in order to obtain relevant data for the composition process. The process of composition may be either automatic or user-guided. When the system is not able to resolve ambiguities in the composition, user assistance is required. To validate the composition plan and to select the optimal composition alternative, the composition executor composite deploys and compares binding possibilities

ACKNOWLEDGEMENT

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

REFERENCES

1. M. M. Theimer and B. N. Schilit, **Disseminating active map information to mobile hosts**, *IEEE Netw.*, 1994.
2. S. Almutairi, **Review on the security related issues in context aware system**, *Int. J. Wirel. Mob. Networks*, 2012.
3. R. Alawadhi and T. Hussain, **A method toward privacy protection in context-aware environment**, in *Procedia Computer Science*, 2019.
4. W. Liu, X. Li, and D. Huang, **A survey on context awareness**, *International Conference on Computer Science and Service System, CSSS 2011 - Proceedings*, 2011.
5. H. Xiao, Y. Zou, J. Ng, and L. Nigul, A**n approach for context-aware service discovery and recommendation**, *IEEE 8th International Conference on Web Services*, 2010.
6. G. M. Kapitsaki, G. N. Prezerakos, N. D. Tselikas, and I. S. Venieris, C**ontext-aware service**

engineering: A survey, *J. Syst. Softw.*, 2009.

7. 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.
8. F. Marton and S. Booth, ***Learning and Awareness,*** 2013.
9. G. Salvaneschi, C. Ghezzi, and M. Pradella, **Context-oriented programming: A software engineering perspective**,*J. Syst. Softw.*, 2012.
10. B. Henderson-Sellers, **Bridging metamodels and ontologies in software engineering**, *J. Syst. Softw.*, 2011.
11. P. R. Lumertz, L. Ribeiro, and L. M. Duarte, **User interfaces metamodel based on graphs,** *J. Vis. Lang. Comput.*, 2016.
12. K. Verbert *et al.*, **Context-aware recommender systems for learning: A survey and future challenges**, *IEEE Transactions on Learning Technologies*, 2012.
13. A. 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.
14. C. Emmanouilidis, R. A. Koutsiamanis, and A. Tasidou, **Mobile guides: Taxonomy of architectures, context awareness, technologies and applications**, *Journal of Network and Computer Applications*. 2013.
15. Y. H. Feng, T. H. Teng, and A. H. Tan, **Modelling situation awareness for context-aware decision support**, *Expert Syst. Appl.*, 2009.
16. R. Welke, R. Hirschheim, and A. Schwarz, **Service-oriented architecture maturity**, *Computer (Long. Beach. Calif.)*, 2011.
17. H. Petritsch, **Service-oriented architecture (SOA) vs . component based architecture**, *Vienna Univ. Technol. white Pap. available …*, 2006.
18. D. 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.
19. S. Kalasapur, M. Kumar, and B. A. Shirazi, **Dynamic service composition in pervasive computing**, *IEEE Trans. Parallel Distrib. Syst.*, 2007.
20. T. G. Stavropoulos, D. Vrakas, and I. Vlahavas, **A survey of service composition in ambient intelligence environments**, *Artificial Intelligence Review*, 2013.
21. K. Fujii and T. Suda, **Semantics-based context-aware dynamic service composition**, *ACM Trans. Auton. Adapt. Syst.*, 2009.
22. 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.
23. 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.
24. IBRAHIM, Noha et MOUEL, Frederic Le. **A survey on service composition middleware in pervasive environments**. *arXiv preprint arXiv:0909.2183*, 2009.