PRAGUE UNIVERSITY OF ECONOMICS AND BUSINESS

*Acta Informatica Pragensia*
2022, Volume 11, Issue 1, pp. 80–104
https://doi.org/10.18267/j.aip.174

**Article** Open Access

# CA-BPEL: A New Approach to Facilitate the Development and Execution of Context-Aware Service Orchestrations

**Hossein Moradi [1], Bahman Zamani [2], Kamran Zamanifar [3]**

[1] Department of Computer Engineering, Birjand University of Technology, Birjand, Iran
[2] MDSE Research Group, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran
[3] Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran

Corresponding author: Bahman Zamani (zamani@eng.ui.ac.ir)

## Abstract

The proliferation of smartphones and sensor-based networks has led to a greater need for context-aware applications and pervasive business processes. One of the key approaches that seek to satisfy this need is context-aware service composition. Service composition can be achieved in two ways, i.e., service choreography and service orchestration. Embedding the context into an orchestrated composite service enhances its flexibility, but makes its development and execution more complicated. This study aims to reduce this complexity by introducing the CA-BPEL approach. Our proposed approach enables developers to turn a standard orchestrated service into a context-aware orchestrated service, consistent with the standard WS-BPEL language. This study applies the Design Science Research Methodology, in which we evaluate CA-BPEL by using a tourism demonstration along with the conduction of a usability survey that shows the convenience of the proposed approach. We also compare our proposed approach with 14 related studies. Our investigations suggest that CA-BPEL has much potential to facilitate the development and execution of context-aware service compositions.

## Keywords

Context-aware service orchestration; CA-BPEL approach; PCASO middleware; Service composition; WS-BPEL language.

**Citation:** Moradi, H., Zamani B., & Zamanifar, K. (2022). CA-BPEL: A New Approach to Facilitate the Development and Execution of Context-Aware Service Orchestrations. *Acta Informatica Pragensia*, 11(1), 80–104. https://doi.org/10.18267/j.aip.174

**Academic Editor:** Zdenek Smutny, Prague University of Economics and Business, Czech Republic

## 1 Introduction

The proliferation of smartphones and advances in technologies such as RFID and sensor-based networks have led to an increase in the popularity of context-aware applications and pervasive business processes (Baldauf et al., 2007; Sheng et al., 2014). Context-aware applications and processes exploit contextual data and information associated with the user and environment entities to automate user activities or make better decisions (Faieq et al., 2021). There exists a wide range of contextual information, e.g., the user's geographical and temporal situation, the user's mood, the user's device battery status, available mobile networks, and the weather status (Moradi et al., 2020; Yigitbas et al., 2020).

Context plays an increasingly important role in business process management (BPM) literature (Vom Brocke et al., 2021). Considering context awareness is critical for managing business processes successfully (Janiesch and Kuhlenkamp, 2019; Vom Brocke et al., 2021). Gartner (2019) identified real-time situational awareness as an essential capability of next-generation intelligent BPM suites (Gartner, 2019; Zhao et al., 2021). Kerpedzhiev et al. (2021) explored the capabilities of future BPM systems through a Delphi study and emphasized that process context management (including context specification, detection, monitoring and change handling) is a central capability that should be covered by future BPM systems (Kerpedzhiev et al., 2021; Vom Brocke et al., 2021).

According to Gartner (2022), composable applications are among the top 12 strategic technology trends in 2022. The composition of services is one of the common strategies for implementing business processes (Nikoo et al., 2020). Service composition, which refers to a combination of multiple services into a more extensive composite service, is classified into choreography and orchestration (Papazoglou and van den Heuvel, 2007). Service orchestrations can be implemented using "standard executable business process languages" such as Business Process Execution Language (BPEL) and Business Process Model and Notation (BPMN) (Lübke and Pautasso, 2019). WS-BPEL, which is a popular standard language to specify enterprise-wide service orchestrations (Botangen et al., 2020; Nikoo et al., 2020), is the "first standard language, emphasizing the fully executable aspect of process models" (Lübke and Pautasso, 2019).

Pervasive context-aware service composition (CASC) enables users to employ dynamic yet composed web services that take contextual information into account (Zhou et al., 2011). In the service composition domain, context refers to any information associated with service users, service requesters, service providers, or the execution environment that may affect the composite service execution (Laleh, 2018). A tourism context-aware recommendation service is an example of a real-world CASC that recommends a customized tourism programme according to the contextual information of tourists, tourism service providers and weather conditions (Baidouri et al., 2012).

Embedding the context into an orchestrated service enhances the business process flexibility (Vom Brocke et al., 2021; Zhou et al., 2011); however, the development and execution of context-aware service composition may be practically complicated and time-consuming. The reasons include diversity and heterogeneity of context concepts and sources, the need to comply with the service composition standards, the temporal and uncertain nature of context values, and the limited resources (such as memory and battery life) of the devices running the applications (Moradi et al., 2020; Sheng et al., 2014). Accordingly, this paper aims to introduce a technique to address the complexity of CASC development and execution.

To achieve this goal, we propose to follow a model-driven development (MDD) approach to generate WS-BPEL-compliant service compositions that are fully compatible with standard service composition execution engines (Hagin, 2011; Ibrahim, 2012). Although the MDD technologies seem to produce promising results for standard service compositions, we may face various challenges when applying them in the CASC domain, since they do not consider contextual information while service compositions are developed or executed (Ibrahim, 2012). Some studies (Furno and Zimeo, 2014; Kocurova, 2013; Schefer-Wenzl and Strembeck, 2013) have used a proprietary solution, inconsistent with industry-accepted standards, making their acceptance a severe challenge. Several existing techniques (Cherif et al., 2016; Furno and Zimeo, 2014; Hagin, 2011) have also embedded contextual information into composite services. For example, Hagin (2011) provided a CASC approach, in which context is exposed as an independent web service, and a WS-BPEL-compliant composite service is integrated into a context service. Our investigations revealed that related studies either support limited dimensions of contextual concepts or have weaknesses in embedding context awareness into a standard composite service. Additionally, no experimental confirmation is reported that the current techniques can reduce the complexity of CASC development and execution.

To resolve the limitations of existing studies, this paper proposes a new solution to facilitate the CASC development and execution. In the first step of this research, we investigated existing studies in the CASC domain and proposed 13 requirement specifications (R1 to R13). To meet R1 to R13, we followed both MDD and Context-as-a-Service (CaaS) approaches and presented a new approach called CA-BPEL (Context-Aware Business Process Execution Language). The CA-BPEL approach facilitates CASC development and execution by introducing a new middleware called PCASO (Pervasive Context-Aware Service Orchestration). CA-BPEL also utilizes standard WS-BPEL-compliant tools along with our previous framework, called CaaSSET (Context-as-a-Service Set of Engineering Tools) (Moradi et al., 2020).

In the CaaSSET work (Moradi et al., 2020), we elaborated on a novel MDD framework in the CaaS domain, along with its five components. CaaSSET (Moradi et al., 2020) focused on modelling, developing and delivering context as a web service. CaaSSET is related to the context service provider layer. However, this research focuses on the context service consumer layer and provides a new solution to facilitate CASC development and execution. Hence, the main contributions of this paper are as follows:

- **Requirements R1 to R13:** By reviewing existing works in the CASC domain, an initial list of requirement specifications is proposed. This list is revised during the development of the CA-BPEL approach. Finally, we propose 13 requirement specifications (R1 to R13) for CASC development and execution.
- **CA-BPEL approach:** CA-BPEL leverages both MDD and CaaS approaches to facilitate CASC development and execution. Our proposed approach is a non-invasive one that incorporates context awareness features into a WS-BPEL model without altering the original meta-model of the WS-BPEL language.
- **PCASO middleware:** It consists of a reference model and five intermediate web services. PCASO enables developers to turn a standard composite service into a WS-BPEL-compliant context-aware composite service. PCASO also facilitates the interoperability of standard composite services and context web services.

To evaluate CA-BPEL, we compare it with 14 baseline techniques. We also show the usefulness of CA-BPEL in the context of CASC by conducting a demonstration in the tourism domain. The judgment on the convenience of CASC development and execution is entirely dependent on satisfying the ease of context service development and delivery by the CaaSSET toolset. Hence, we evaluate this toolset by conducting a usability survey.

The remaining sections of this paper are organized as follows. In Section 2, the methodology and scope of our research are explained. Section 3 presents the research background and literature. Section 4 introduces a list of 13 requirement specifications. Our proposed solution to satisfy the identified requirements is explained in Section 5. Section 6 shows the advantages and usefulness of the CA-BPEL approach by conducting a tourism demonstration, along with a usability evaluation survey. Finally, Section 7 concludes the paper and presents some ideas for future work.

## 2 Research Method and Scope

**Research methodology:** The methodology of this research is based on the Design Science Research Methodology (DSRM) presented by Peffers et al. (2007). Figure 1 shows the six steps of the mentioned methodology and the techniques used in each step.

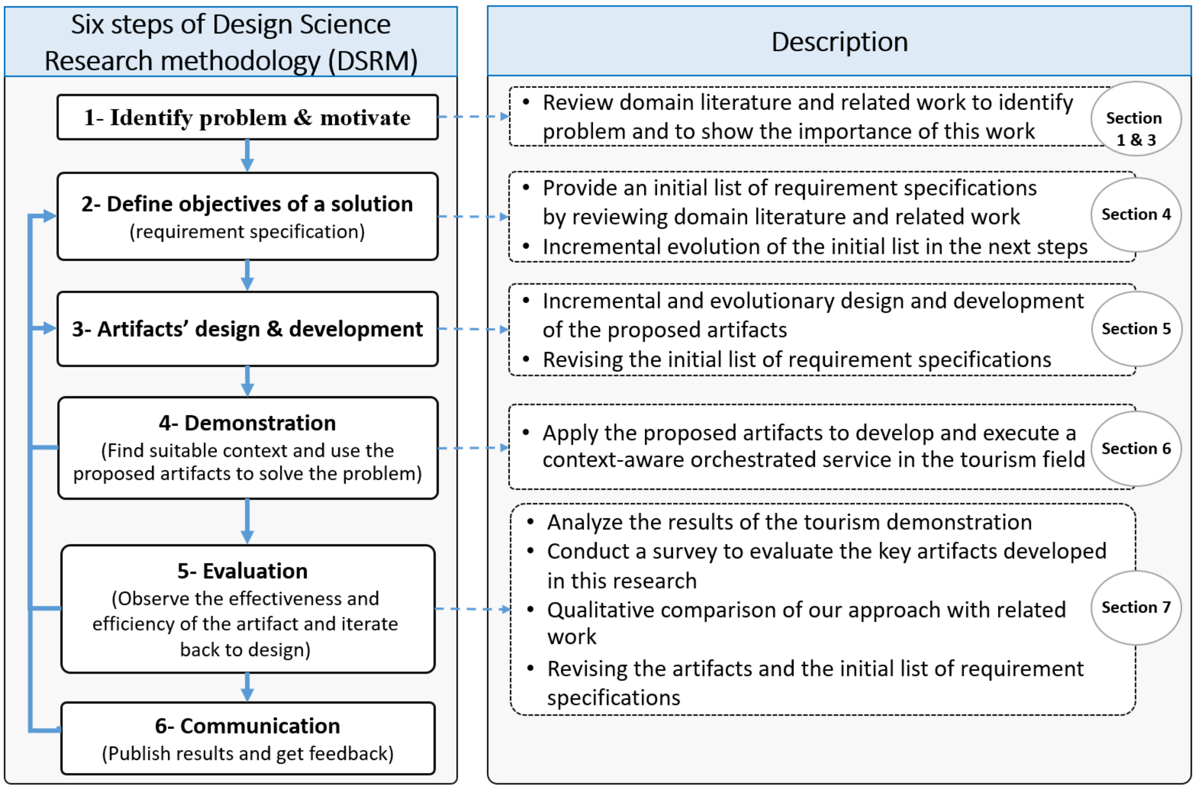
Six steps of Design Science Research methodology (DSRM)
Description
1- Identify problem & motivate
2- Define objectives of a solution (requirement specification)
3- Artifacts' design & development
4- Demonstration (Find suitable context and use the proposed artifacts to solve the problem)
5- Evaluation (Observe the effectiveness and efficiency of the artifact and iterate back to design)
6- Communication (Publish results and get feedback)
• Review domain literature and related work to identify problem and to show the importance of this work
Section 1 & 3
• Provide an initial list of requirement specifications by reviewing domain literature and related work
• Incremental evolution of the initial list in the next steps
Section 4
• Incremental and evolutionary design and development of the proposed artifacts
• Revising the initial list of requirement specifications
Section 5
• Apply the proposed artifacts to develop and execute a context-aware orchestrated service in the tourism field
Section 6
• Analyze the results of the tourism demonstration
• Conduct a survey to evaluate the key artifacts developed in this research
• Qualitative comparison of our approach with related work
• Revising the artifacts and the initial list of requirement specifications
Section 7

***Figure 1:*** *Six steps of Design Science Research Methodology (DSRM). Source: Peffers et al. (2007)*

Our methodology begins in the first step with the explanation of the research problem and identifying the challenges related to CASC development and execution. In the second step, an initial list of requirement specifications is provided. In the third step, the proposed research solution is designed and implemented based on the initial requirement list. Note that our proposed research artifact is revised according to the evolutional and incremental development method, and after several revision stages, we present the final version of the proposed artifact. In the fourth step, we use our proposed artifact to develop a context-aware composite service in the field of tourism that can be executed on a standard service composition engine. In the fifth step, the proposed research solution is evaluated. In the final step, conclusions are made, suggestions for future work are presented and research results and achievements are published.

**Research scope:** Service composition is classified into service choreography and service orchestration. In service choreography, there exist equivalent services that can communicate together without any central coordinator service. In contrast, in the service orchestration approach, there is a coordinator service that orchestrates the underlying services (Papazoglou and van den Heuvel, 2007). WS-BPEL is a de facto standard language to specify enterprise-wide service orchestrations (Botangen et al., 2020). Due to the various applications of service orchestration in the management of business processes and the widespread acceptance of the WS-BPEL standard by industries and universities (Lübke and Pautasso, 2019, Botangen et al., 2020; Nikoo et al., 2020), this research focuses on WS-BPEL-compliant context-aware service orchestration. Thus, context-aware service choreography is out of the research scope and may be the subject of future research. Hence, whenever this paper refers to service composition, it means service orchestration.

## 3 Background

This section briefly reviews the research background and literature, including contextual data and information, CASC, CaaS and MDD.

### 3.1 Contextual data and information

Salber et al. (1999) defined context as "*any information that can be used to characterize the situation of an entity*". They also defined the entity (context-aware object) as an object, person or place relevant to a user's interaction with an application.

Context is classified into two general categories: primary and secondary (Perera et al., 2014). Primary context refers to context elements (contextual data), and secondary refers to high-level contexts (contextual information) (Moradi et al., 2020). Moradi et al. (2020) defined the context element as a set of context attributes, dealing with low-level contextual data and the high-level context as "*the state of a context-aware object (entity) from a specific perspective*". For example, data received from GPS sensors (e.g., longitude: 5677183.0 and latitude: 51.528308) is the context element, while the geographical location (e.g., the current city: London) that is inferred by data processing is the high-level context (Perera et al., 2014; Moradi et al., 2020).

Note that context information is not limited to the geographic location and includes various aspects such as weather situation (e.g., sunny, rainy, snowy or stormy), time of day (e.g., morning, noon, evening or midnight), available mobile networks (e.g., 2G, 3G, 4G or 5G), the user's device battery level (e.g., low, medium or high), the user's mood (e.g., normal, happy or sad), or the ambient light level (e.g., dim, normal or bright) (Salber et al., 1999; Hoyos et al., 2016; Moradi et al., 2020; Yigitbas et al., 2020).

### 3.2 Context-aware service composition (CASC)

Combining several services to build a composite service is called service composition. The resulting composite service can be re-composed with other services to create more abstract yet useful composite services (Papazoglou and van den Heuvel, 2007). The proliferation of context-aware applications has increased the need for CASC development. In CASC, contextual information is considered while services are composed. This kind of composite service will be more flexible than standard composite services since the service composition adapts itself to the user entity's current situation (Zhou et al., 2011).

### 3.3 Context-as-a-Service (CaaS)

Developing a standalone context-aware application that manages all aspects of the context model is costly, time-consuming and complex (Chabridon et al., 2013; Moore et al., 2014; Vahdat-Nejad, 2014). Hence, the CaaS idea has been proposed, which enables developers to focus on consuming context services instead of developing context management components from scratch. In this approach, context service provision is outsourced to third-party context service providers (Hynes et al., 2009; Moore et al., 2014; Yigitbas et al., 2020). Several approaches, such as CoaaS (Hassani et al., 2018), CDQL (Hassani et al., 2019), NGSI (Open Mobile Alliance, 2012), NGSI-LD (ETSI, 2019), and CaaSSET (Moradi et al., 2020), have proposed solutions and architectures to facilitate CaaS delivery. In our previous work on CaaSSET (Moradi et al., 2020), we suggested four improvements to the CoaaS (Hassani et al., 2018) architecture for achieving a lightweight CaaS platform with the least dependence on the context broker layer. Figure 2 shows that the proposed architecture is divided into four main layers: context service providers, context service consumers, context brokers and context sources (Moradi et al., 2020).

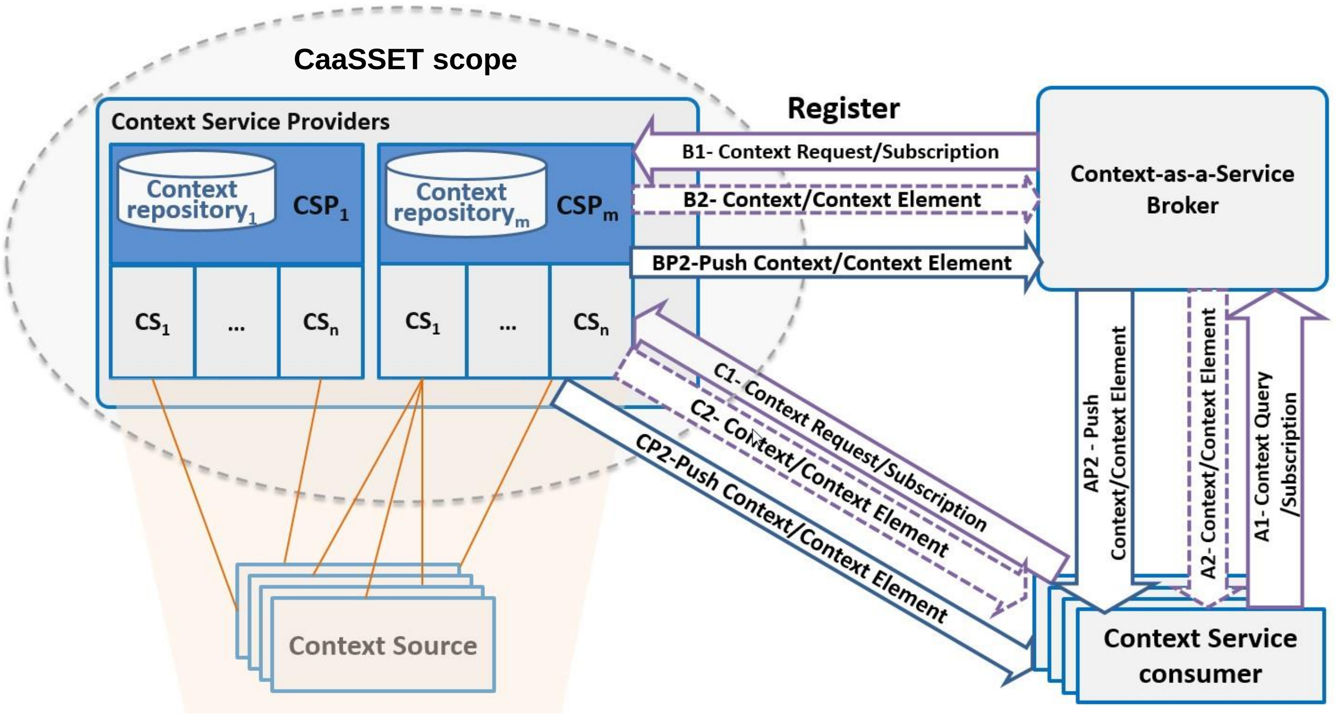
CaaSSET scope
Context Service Providers
Context repository$_1$
CSP$_1$
Context repository$_m$
CSP$_m$
CS$_1$
...
CS$_n$
CS$_1$
...
CS$_n$
Context Source
Register
B1- Context Request/Subscription
B2- Context/Context Element
BP2-Push Context/Context Element
Context-as-a-Service Broker
C1- Context Request/Subscription
C2- Context/Context Element
CP2-Push Context/Context Element
AP2 - Push Context/Context Element
A2- Context/Context Element
A1- Context Query /Subscription
Context Service consumer

***Figure 2:*** *Lightweight CaaS architecture. Source: Moradi et al. (2020)*

In this architecture (Moradi et al., 2020), each context service provider may host and manage one or more context services and deliver each context service as an autonomous context web service. The context source is an optional layer located below the context service provider layer that provisions the values of context elements for one or more context services. Context sources are classified into four general categories: sensor, middleware, web service and profiled (repository). In the proposed architecture, we redefined the context service scope to cover the key contextual concepts of the context service and their associated relationships and features. In our terminology, "*context service is an interface that abstracts its underlying context sources and provides context elements and high-level contexts, which are related to one or more context-aware objects (entities)*" (Moradi et al., 2020).

## 3.4 Model-driven development (MDD)

To facilitate context service development, we can leverage various approaches, one of which is the MDD approach (Chabridon et al., 2013). In software engineering, abstraction is a well-known method to overcome complexity. MDD considers models as first-class citizens and uses a higher layer of abstraction in software development (Bézivin, 2004; Brambilla et al., 2012). In the MDD approach, we typically create a domain-specific language (DSL) that focuses on resolving problems in a particular domain. A DSL has significant potential to increase the productivity of its users because it is designed to facilitate the modelling and development of specific requirements (Brambilla et al., 2012).

The first prerequisite for developing a DSL is to define the key concepts and relationships of the problem domain as a meta-model (Bézivin, 2004). In addition to the definition of an abstract syntax for a DSL, we should define the layout and symbols of our modelling tools, called concrete syntax (Brambilla et al., 2012). The MDD approach allows developers to design a model using a textual or graphical modelling tool. This model is then (semi-)automatically transformed into executable code using model transformation techniques (Brambilla et al., 2012).

### 3.5 MDD in self-adaptive context-aware applications

Recently, various studies applied the MDD approach for developing complex context-aware applications, which can adapt themselves to the context of use. For instance, in the area of human-computer interaction (HCI), studies such as Yigitbas et al. (2020) and Abrahão et al. (2021) have used MDD to facilitate the development of context-aware self-adaptive user interfaces (SAUIs). Yigitbas et al. (2020) proposed a new approach, in which context information is modelled and transformed into a context service. The user interface (UI) model and the adaptation model are also defined and transformed into an executable UI and adaptation service, respectively. This approach finally allows runtime user interface adaptation based on the latest contexts of the user, environment and usage platform.

## 4 Requirement List

By reviewing existing works in the CASC domain, an initial list of requirement specifications was proposed. Note that this initial list of requirements has been prepared with the assumption that our proposed solution uses the Context-as-a-Service (CaaS) approach. This initial list was gradually revised during the development of CA-BPEL and finally, we proposed 13 requirement specifications (R1 to R13) for CASC development and execution. Table 1 shows the 13 requirement specifications classified in three categories: service composition and execution, embedding context awareness into composite services, and context service development and delivery.

***Table 1:** List of requirement specifications for CASC development and execution*

| Classification | CASC Requirement Specifications (R1 to R13) |
|---|---|
| **Service composition and execution** | **R1:** Supporting the development of standard service composition models (service orchestration) |
| | **R2:** Compatibility of composite services with standard service composition execution engines |
| **Embedding context awareness into composite services** | **R3:** Push notification context trigger: Initiate a composite service by a context service as soon as a specific context is satisfied |
| | **R4:** Periodic pull context trigger: Starting a composite service by periodically pulling the satisfaction status of a specific context from a context service |
| | **R5:** Ability to define the synchronous or asynchronous contextual pre/post-condition for each action of a composite service |
| | **R6:** Interoperability of composite services with various web service technologies (especially REST and SOAP) |
| | **R7:** Compatibility of CASC with standard service composition execution engines |
| | **R8:** Ease of CASC development |
| **Context service development and delivery** | **R9:** Context delivery as a web service, independently of a composite service |
| | **R10:** Sharing of context values with multiple context-aware composite services (via a standard API) |
| | **R11:** Easy run-time access to current and historical context values (via a standard API) |
| | **R12:** Support for a comprehensive context service model |
| | **R13:** Ease of context service development and delivery |

## 5 Proposed CA-BPEL Approach

The CA-BPEL approach enables designers to develop and deliver context web services independent of composite services. CA-BPEL enables developers to build a context-aware composite service compatible with standard service composition engines. In the CA-BPEL approach, the contextual concepts are not embedded into the meta-model of a standard service composition language (such as WS-BPEL). Instead, the context concepts are exposed as a context web service, and service compositions call the context web service similar to other standard web services. Figure 3 depicts the general architecture of the CA-BPEL approach. CA-BPEL consists of three components: the PCASO middleware, the CaaSSET framework (Moradi et al., 2020) and standard WS-BPEL-compliant tools. As seen in Figure 3, the position of these three components (i.e., CaaSSET, PCASO and standard WS-BPEL tools) are marked from left to right with the letters A, B and C, respectively.

The main audience of CA-BPEL is business process developers who want to build context-aware business processes by embedding context awareness into standard service orchestrations. They are experts in modelling and developing WS-BPEL-compatible service orchestrations, but aim to avoid the complexities associated with the development of a context service and the difficulties involved in integrating a context service with a standard service orchestration.

In CA-BPEL, we use free standard WS-BPEL tools available on the market to develop and execute standard composite services. Developers can use the standard WS-BPEL development tools to model and build a composite service that reuses the context web service generated by the CaaSSET framework. Developers can also use the new proposed PCASO middleware to embed the interoperability and context awareness features into a standard composite service. By leveraging the CA-BPEL approach, developers can build a WS-BPEL-compliant context-aware composite service that is executable on standard service composition engines.

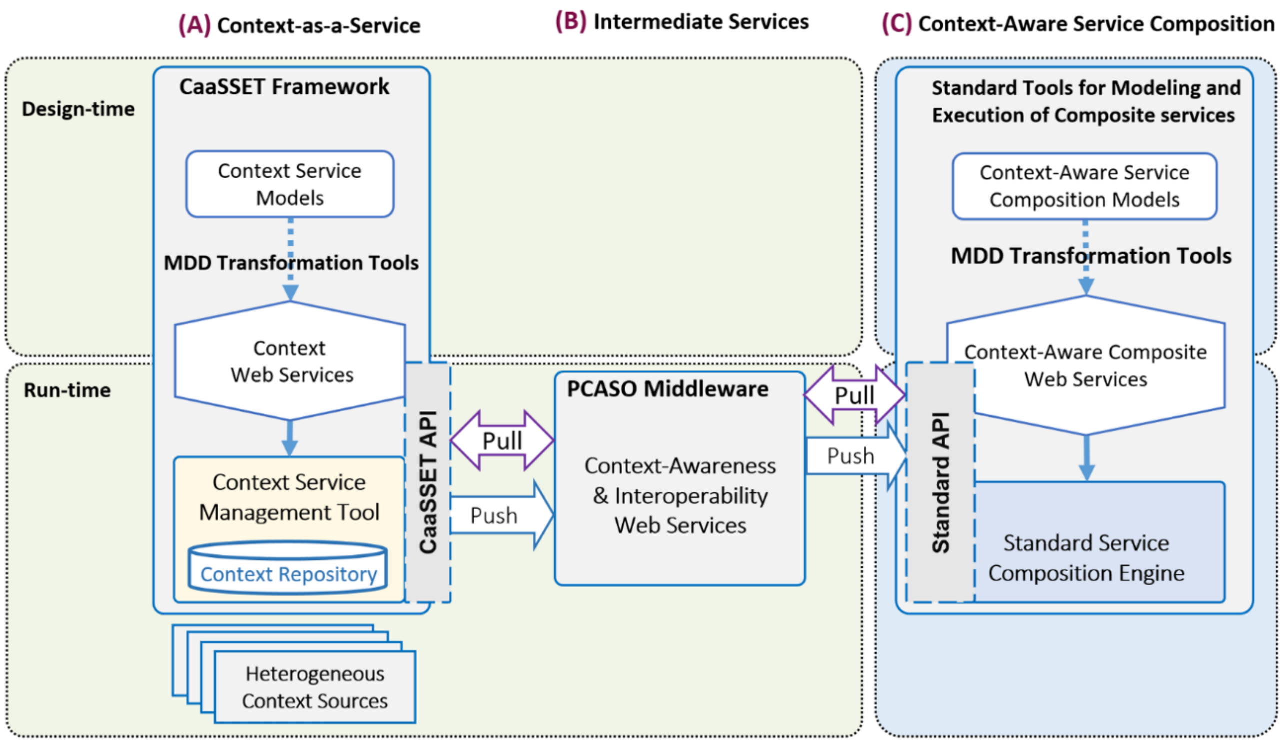
(A) Context-as-a-Service
(B) Intermediate Services
(C) Context-Aware Service Composition
Design-time
Run-time
CaaSSET Framework
Context Service Models
MDD Transformation Tools
Context Web Services
Context Service Management Tool
Context Repository
CaaSSET API
Heterogeneous Context Sources
Pull
Push
PCASO Middleware
Context-Awareness & Interoperability Web Services
Pull
Push
Standard API
Standard Tools for Modeling and Execution of Composite services
Context-Aware Service Composition Models
MDD Transformation Tools
Context-Aware Composite Web Services
Standard Service Composition Engine

*Figure 3: Big picture of CA-BPEL approach*.

We introduced the CaaSSET framework in our previous work (Moradi et al., 2020). CaaSSET consists of a reference model, a meta-model and a toolset, containing a graphical modelling tool, code generation tools and an API. Developers can use the CaaSSET toolset to graphically model a context service and transform it into an executable context web service. The CaaSSET API is equipped with a context repository and allows context delivery as a REST web service and run-time access to the context values.

It should be noted that the public artifacts of CA-BPEL, such as the CaaSSET Eclipse plug-in, are available at the GitHub repository[1] of CA-BPEL. The context web service of TourismCS, generated by the CaaSSET framework (Moradi et al., 2020), can also be accessed via the GitHub repository.

## 5.1 PCASO middleware

The PCASO middleware allows developers to simply turn a standard composite service into a context-aware composite service. PCASO is fully compatible with the standard WS-BPEL language. WS-BPEL is typically used to model composite services and is known as the accepted standard of service composition execution engines. PCASO also facilitates the interoperability of standard composite services and context web services.

The PCASO middleware consists of a reference model and five intermediate web services. We depicted the PCASO reference model in Figure 4. The reference model refers to partitioning the functions into several elements and drawing the data flow of each element (Armstrong, 2014). The PCASO web services include a context evaluation service, a context service adapter, context trigger services (a periodic pull service and a push notification service) and a context listening service.

### 5.1.1 PCASO context service adapter

A context service is typically implemented using the SOAP or REST technologies. In the PCASO reference model, instead of establishing a direct connection between a composite service and a context service, these services are connected through the PCASO context service adapter. PCASO provides these adapters as a SOAP web service. This adapter resolves the interoperability according to the invoked context web service type (e.g., SOAP or REST).

### 5.1.2 PCASO context evaluation service

A context evaluation service receives a context value and compares it with predefined values. If they match, the value of one is returned, and otherwise, the value of zero is returned. Predefined values can be either a fixed value, a list of allowed values or a range of allowed values limited by a lower and an upper bound. A context evaluation service is used by three PCASO web services (i.e., the context listening service, the periodic pull service and the push notification service).

[1] See https://github.com/hsmoradi/ca-bpel/wiki

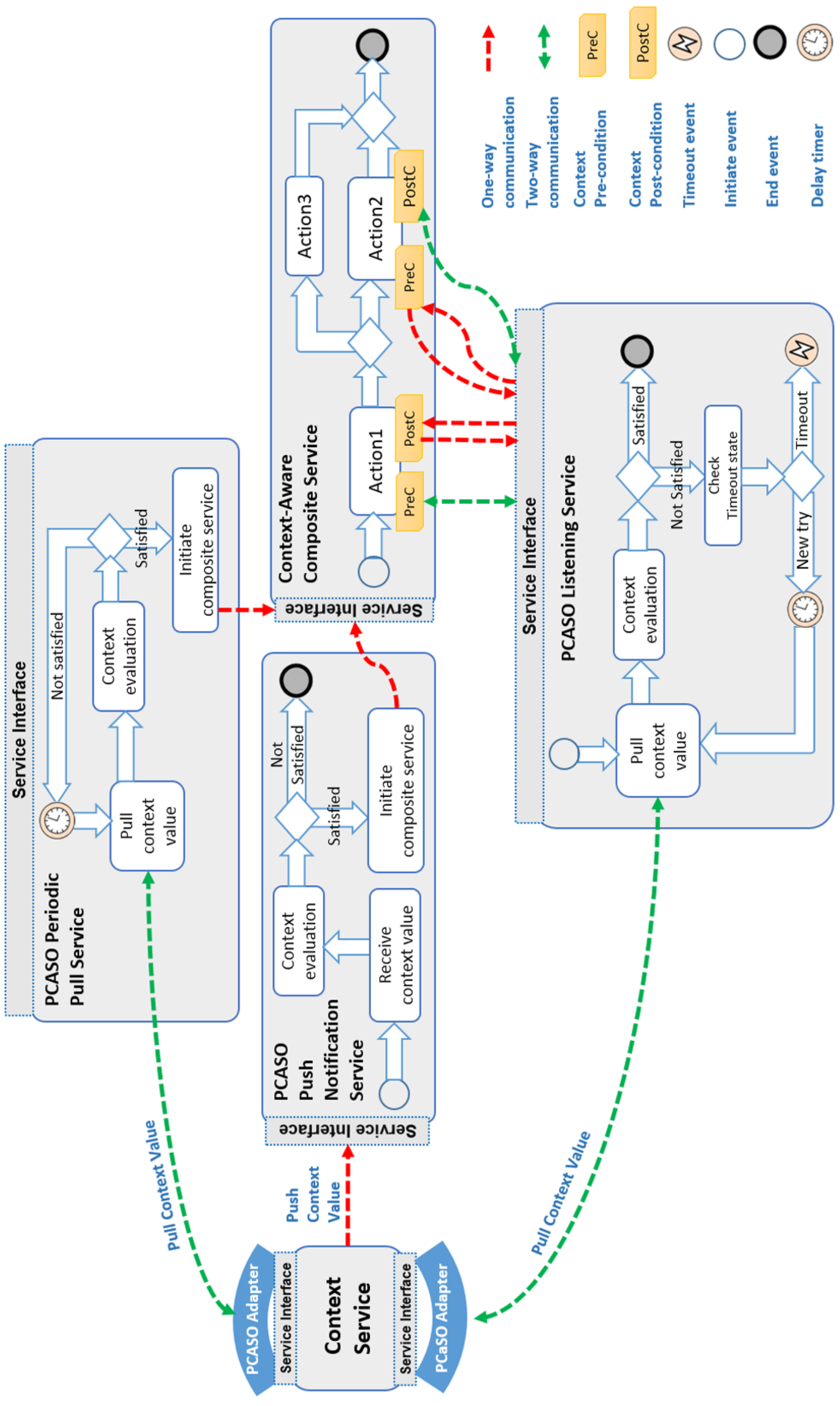

***Figure 4:** PCASO reference model.*

For embedding context awareness into a standard composite service, PCASO introduces two mechanisms: a context trigger and context listening.

### 5.1.3 PCASO context trigger

When we activate a context trigger, the context value is monitored and evaluated. When a context value matches the predefined values, the trigger is activated and a predefined composite service is called and initiated. The context triggers can be activated by either of the two web services provided by PCASO, i.e., the periodic pull service or the push notification service.

Periodic pull service: A predefined service is executed periodically (e.g., every 5 seconds) and extracts the last value of a specific context from a particular context service. The extracted context value is evaluated, and if it matches the predefined values, the trigger is activated. If the extracted value does not satisfy the evaluation conditions, the periodic pull service snoozes for a pre-determined period (e.g., 30 seconds). After the snooze, the cycle of pulling and evaluating context values is repeated periodically.

Push notification service: The composite service is subscribed in advance for a specific context of a particular context service. Afterwards, the context service monitors the context values. As soon as it detects any change in the monitored context, its new context value is sent to the push notification service. This service evaluates the received context value. In the case of compliance, the trigger is activated. Note that the context service must support the push notification feature to support the PCASO push notification service fully. Fortunately, this feature is supported by the CaaSSET API.

### 5.1.4 PCASO listening service

PCASO uses the existing features of WS-BPEL to add the contextual pre-condition and contextual post-condition before or after each action, respectively. When a service composition is executed, PCASO initiates the listening service right after reaching any contextual pre/post-condition. By calling the PCASO listening service, the last context value is extracted from the associated context service. This context value is evaluated. When it matches with predefined values, the value of one is returned to the composite service. It means that the contextual pre/post-condition is satisfied. In the case of non-compliance, the listener snoozes for a predefined period (e.g., 30 seconds). Afterwards, the last context value is re-extracted and re-evaluated. This cycle is repeated until the contextual pre/post-condition is satisfied or the listening deadline is reached. As soon as the deadline condition is reached, the listening service is stopped and the negative value of one is returned to the composite service.

The PCASO listening service can be called both synchronously and asynchronously. In synchronous listening, the composite service suspends until it receives a response from the PCASO listening service or reaches the listening deadline. Asynchronous listening means that the composite service sends a listening request and continues the service composition. In this approach, the calling composite service is not suspended. Note that synchronous listening is suggested when the listening deadline is short-term (e.g., less than 30 seconds); otherwise, asynchronous listening is recommended.

## 5.2 Standard tools for developing and executing composite services

We use the standard Eclipse BPEL Designer[2] tool to develop composite service models and transform them into WS-BPEL code. The standard Eclipse BPEL Designer tool provides a standard graphical modelling environment, enabling developers to build a WS-BPEL-compliant composite service model and integrate it with its constituting web services. This development environment automatically generates the equivalent WS-BPEL code of the graphical service composition model. Our proposed solution also utilizes the WSO2 BPS (Business Process Server) tool[3] to deploy and execute composite services. This open-source

[2] See https://www.eclipse.org/bpel/

[3] See https://docs.wso2.com/display/BPS360/

tool facilitates the deployment and management of composite services. Note that WSO2 BPS uses the standard Apache ODE (Orchestration Director Engine)[4] to execute composite services.

To fully meet requirements R1 to R13, this study used the de-facto standard WS-BPEL language and its associated tools to implement and execute the composite services. However, the conceptual design of the proposed approach does not depend much on this standard. On the other hand, our main contributions (i.e., CA-BPEL, CaaSSET and PCASO) were designed according to the separation of concerns and reusability principles. Therefore, in future works, we can customize CA-BPEL for embedding context awareness into service choreography standards, the BPMN standard, or non-standard commercial service composition platforms, such as Amazon AWS Step Functions and Netflix Conductor. According to our preliminary investigations, such customizations can be made with minor modifications to the CaaSSET framework, which is the core of the CA-BPEL approach.

## 6 Demonstration (Tourism Context-aware Composite Service)

In our previous research (Moradi et al., 2020), we conducted a tourism demonstration and utilized the CaaSSET toolset to model, develop and deliver the TourismCS context service. In this section, we conduct a tourism demonstration to show the usefulness of CA-BPEL in the CASC domain. This demonstration can model a tourism composite service using standard service composition modelling tools. Afterwards, we use the PCASO adapter to facilitate the interoperability of OutdoorTourProcess and the TourismCS context service.

We also used the PCASO web services to embed the context awareness features into the tourism composite service. Finally, the CA-BPEL approach could generate a WS-BPEL-compliant context-aware composite service, which can be executed on the Apache ODE service composition engine.

We illustrated the conceptual model of OutdoorTourProcess in Figure 5. This figure shows that a standard composite service consists of two atomic web services (i.e., TourOrganizerWS and TouristNotifierWS) and one context service (i.e., TourismCS). The communication types of OutdoorTourProcess and the two atomic web services are one-way asynchronous and two-way synchronous, respectively.

In Figure 5, we followed the CA-BPEL approach and created a context-aware service composition by adding green elements to a standard composite service. The white colour was also used for the main elements of the standard composite service. Similarly, the atomic services, which are reused by the composite service, were marked grey.

To design and develop OutdoorTourProcess, we used the Eclipse BPEL Designer environment. In this IDE, a WS-BPEL-compliant service composition model was drawn graphically, and the communication with the atomic web services was configured. To turn this standard composite service into a context-aware composite service, we manually connected the tourism composite service and the intermediate web services provided by PCASO. We also configured the communication between the PCASO context service adapter and the TourismCS context web service. Eclipse BPEL Designer generated the WS-BPEL code of OutdoorTourProcess automatically. After designing and developing OutdoorTourProcess, we deployed its WS-BPEL code in the WSO2 BPS tool. Note that BPS executes the composite services using the standard Apache ODE engine.

[4] See https://ode.apache.org/

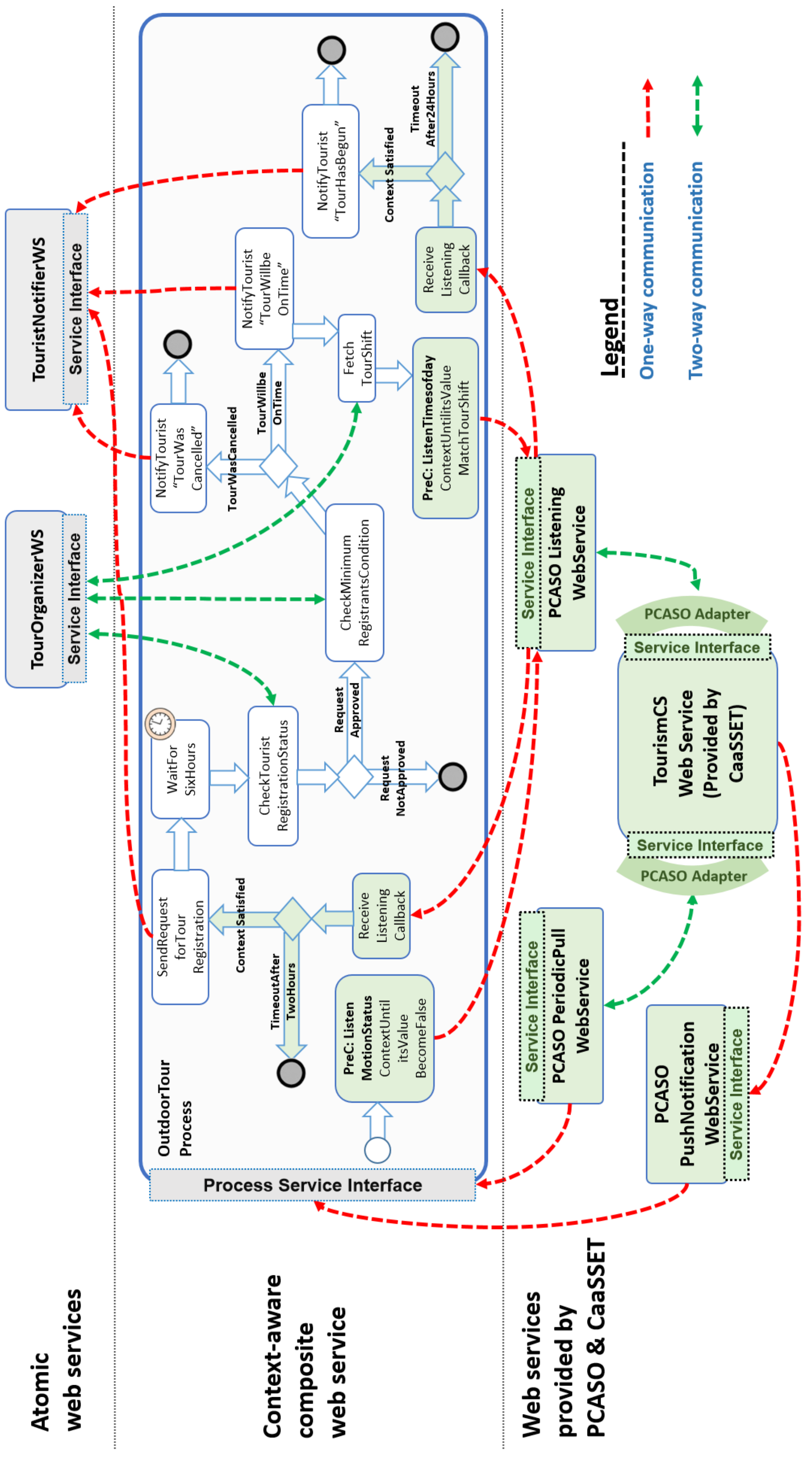
Atomic web services
Context-aware composite web service
Web services provided by PCASO & CaaSSET
TouristNotifierWS
Service Interface
TourOrganizerWS
Service Interface
OutdoorTour Process
Process Service Interface
PreC: Listen MotionStatus ContextUntil itsValue BecomeFalse
TimeoutAfter TwoHours
Context Satisfied
Receive Listening Callback
SendRequest forTour Registration
WaitFor SixHours
CheckTourist RegistrationStatus
Request Approved
Request NotApproved
CheckMinimum RegistrantsCondition
TourWasCancelled
TourWillbe OnTime
NotifyTourist "TourWas Cancelled"
NotifyTourist "TourWillbe OnTime"
Fetch TourShift
PreC: ListenTimesofday ContextUntilitsValue MatchTourShift
Receive Listening Callback
Context Satisfied
Timeout After24Hours
NotifyTourist "TourHasBegun"
Service Interface
PCASO Listening WebService
PCASO Adapter
Service Interface
TourismCS Web Service (Provided by CaaSSET)
Service Interface
PCASO Adapter
Service Interface
PCASO PeriodicPull WebService
PCASO PushNotification WebService
Service Interface
Legend
One-way communication
Two-way communication

*Figure 5:* Conceptual model of tourism CASC (OutdoorTourProcess).

## 7 Evaluation

Section 6.2 introduced a tourism demonstration to evaluate the usefulness of the CA-BPEL approach for CASC development and execution. In this section, the CA-BPEL approach itself is evaluated. In Section 7.1, we use the tourism demonstration results to investigate to what extent PCASO and standard WS-BPEL tools support R1 to R8. In Section 7.2, we discuss to what extent requirements R9 to R13 are supported by the CaaSSET toolset. Note that judgment on the convenience of CASC development and execution is entirely dependent on satisfying the ease of context service development and delivery (R13) by the CaaSSET toolset. Hence, in Section 7.3, we conduct a usability evaluation survey to measure the satisfaction of R13 by the CaaSSET toolset. To resolve threats to the survey validity, we also use a Pearson correlation analysis.

In the continuation of this section, we examine to what extent CA-BPEL supports requirement specifications R1 to R13.

### 7.1 Meeting requirements R1 to R8

In the CA-BPEL approach, we used the standard WS-BPEL tools to develop and execute composite services. Since these tools are more than a decade old and are reasonably mature, they are appropriate for service composition development and execution to the best of our knowledge. Therefore, our proposed solution fully supports both requirements R1 and R2. In the following, we used the tourism demonstration results to examine to what extent PCASO supports requirement specifications R3 to R8.

- The TourismCS context service sends the value of the *OutdoorWeather* context to the PCASO push notification service as soon as it changes. The push notification service evaluates the received context value, and in the case of compliance with the value *true*, the trigger is immediately activated and OutdoorTourProcess is initiated [Requirement R3].
- The PCASO periodic pull service runs periodically and extracts the value of the *OutdoorWeather* context from the TourismCS context service. The PCASO periodic pull service evaluates the *OutdoorWeather* context value. If the context value complies with the value *true*, the trigger is immediately activated and OutdoorTourProcess is initiated [Requirement R4].
- Figure 5 shows that two contextual pre-conditions are evaluated asynchronously by the asynchronous invocation of the PCASO listening service [Requirement R5].
- To support the interoperability between OutdoorTourProcess and TourismCS, both PeriodicPullWebService and ListeningWebService call the PCASO adapter, and this adapter calls the TourismCS service interface on behalf of PeriodicPullWebService and ListeningWebService. This design style allows developers to apply integration without worrying about the interoperability of a SOAP-compliant composite service (i.e., OutdoorTourProcess) and a REST-compliant context service (i.e., TourismCS) [Requirement R6].
- As implied from the tourism demonstration, our approach treats the context model similar to a standard web service. We also have not applied any changes to the original meta-model of WS-BPEL. Besides, our proposed solution builds a context-aware composite service, which can be compatibly executed on the Apache ODE, which is a WS-BPEL-compliant service composition engine [Requirement R7].
- CA-BPEL facilitates CASC development, because, instead of providing a brand new tool for CASC development, we used the Eclipse standard IDE to ease the development of a WS-BPEL-compliant composite service. We also use the CaaSSET framework to facilitate context service development and delivery. Besides, PCASO web services, which are invoked similarly to standard web services, could facilitate turning a tourism composite service into a context-aware tourism composite service [Requirement R8].

## 7.2 Meeting requirements R9 to R13

As mentioned previously, requirements R9 to R13, which are related to context service development and delivery, should be supported by the CaaSSET framework. As implied from our previous work (Moradi et al., 2020), developers can use the CaaSSET toolset to graphically model a context service and transform it into an executable standalone context web service. The CaaSSET API is equipped with a context repository and allows context sharing and delivery as a web service. This web-based REST API supports run-time access to current and historical context values. Hence, CaaSSET fully supports R9 to R11.

Our previous investigations (Moradi et al., 2020) also showed that 34 out of 36 features of the context service modelling are fully supported by CaaSSET, one feature (sophisticated derivation expressions) is partially supported, and one feature (ontology model) is not supported. Note that the ontology model could be incompatible with the vital requirement R13, i.e., the ease of context service development and delivery. These investigations reveal the superiority and comprehensiveness of the CaaSSET meta-model for modelling complex context services compared to the related work. Therefore, we can argue that CaaSSET can fully support R12.

Note that our work aims to ease CASC development and execution. Achieving this goal is entirely dependent on satisfying the vital requirement specification R13 (i.e., ease of context service development and delivery). Hence, meeting R13 by the CaaSSET toolset would be of great value. In the next section, we measure the extent to which the CaaSSET toolset satisfies the critical requirement specification R13.

## 7.3 Measuring the satisfaction of requirement R13 by CaaSSET

Given that the CaaSSET toolset uses a graphical user interface, we used two usability questions to assess the extent to which the CaaSSET toolset supports R13.

1) **First usability question (ease of learning and use):** Is it easy to learn and use the CaaSSET toolset?
2) **Second usability question (audience satisfaction and usefulness):** Is it useful and satisfactorily leveraging the CaaSSET toolset to ease context service development and delivery?

### 7.3.1 CaaSSET toolset usability questionnaire

To answer the usability questions mentioned above, we used the survey method and utilized the Lund questionnaire (Lund, 2001), known as USE. The USE questionnaire measures the usability of the software product interface using the four variables: ease of use, ease of learning, usefulness and audience satisfaction (Lund, 2001). The variables ease of use and ease of learning help find the answer to the first usability question, i.e., *ease of learning and use*. The variables usefulness and audience satisfaction help find the answer to the second usability question, i.e., *audience satisfaction and usefulness*. The questionnaire for evaluating the CaaSSET toolset usability, presented in Table 2, measures the four usability variables based on 11 evaluation factors and 21 questions.

***Table 2:*** *Questionnaire for evaluating CaaSSET toolset usability*

| **Usability dimensions** | | | **Questions and answers** | | | | | | | **Satisfaction ratio** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Usability question** | **Usability variable** | **Usability factor** | **Question** | **No answer** | **Excellent (5)** | **Good (4)** | **Moderate (3)** | **Weak (2)** | **Very weak (1)** | **Good or Excellent (4 or 5)** |
| Easy to learn and use | Easy to use | Easy to work with proposed toolset | 1. Easy to work with the CaaSSET modelling tool | 0% | 41.7% | 58.3% | 0% | 0% | 0% | 100% |
| | | | 2. Easy to work with the CaaSSET code generation tool | 0% | 66.7% | 33.3% | 0% | 0% | 0% | 100% |
| | | | 3. Easy to work with the CaaSSET API | 0% | 50% | 50% | 0% | 0% | 0% | 100% |
| | | User-friendliness | 4. User-friendly modelling tool | 0% | 16.7% | 66.7% | 8.3% | 8.3% | 0% | 83.4% |
| | | | 5. User-friendly code generation tools | 0% | 33.3% | 66.7% | 0% | 0% | 0% | 100% |
| | | Flexibility | 6. Flexibility of modelling and code generation tools to develop the context web service intended by a developer | 0% | 58.3% | 33.4% | 8.3% | 0% | 0% | 91.7% |
| | | | 7. Flexibility of the CaaSSET API for accessing preferred contextual data and information | 0% | 41.7% | 58.3% | 0% | 0% | 0% | 100% |
| | | Consistency and coordination | 8. Consistency and coordination of modelling tool and code generation tools | 8.3% | 58.3% | 16.7% | 16.7% | 0% | 0% | 75% |
| | | | 9. Consistency of CaaSSET API functions | 8.3% | 66.7% | 25% | 0% | 0% | 0% | 91.7% |
| | | Error checking and fast recovery | 10. Error checking and model validation features of the CaaSSET modelling tool | 0% | 50% | 50% | 0% | 0% | 0% | 100% |
| | | | 11. Ability to quickly return to the modelling cycle in terms of detecting a modelling error in code generation tools | 0% | 58.3% | 33.4% | 8.3% | 0% | 0% | 91.7% |
| | | | 12. Displaying appropriate error messages while developers interact with CaaSSET API functions | 8.3% | 41.7% | 50% | 0% | 0% | 0% | 91.7% |
| | Easy to learn | Ease of learning | 13. Easy to learn how to work with the CaaSSET modelling tool | 0% | 58.3% | 33.4% | 8.3% | 0% | 0% | 91.7% |
| | | | 14. Easy to learn how to work with the CaaSSET code generation tools | 0% | 66.7% | 25% | 8.3% | 0% | 0% | 91.7% |
| | | | 15. Easy to learn how to work with the CaaSSET API functions | 0% | 58.3% | 16.7% | 25% | 0% | 0% | 75% |
| Audience satisfaction and usefulness | Usefulness | Applicability | 16. Applicability of the CaaSSET toolset to context service development and delivery | 0% | 50% | 50% | 0% | 0% | 0% | 100% |
| | | Time-saving | 17. Utilizing the CaaSSET toolset for context service development and delivery leads to time savings | 0% | 66.7% | 25% | 8.3% | 0% | 0% | 91.7% |
| | | Effectiveness and productivity | 18. Utilizing the CaaSSET toolset increases the effectiveness and | 8.3% | 41.7% | 41.7% | 8.3% | 0% | 0% | 83.4% |

| Usability dimensions | | | Questions and answers | | | | | | | Satisfaction ratio |
|---|---|---|---|---|---|---|---|---|---|---|
| Usability question | Usability variable | Usability factor | Question | No answer | Excellent (5) | Good (4) | Moderate (3) | Weak (2) | Very weak (1) | Good or Excellent (4 or 5) |
| | | | productivity of context service development and delivery | | | | | | | |
| | | Requirement coverage | 19. The CaaSSET toolset covers important requirements of context service development and delivery | 25% | 25% | 50% | 0% | 0% | 0% | 75% |
| | Audience satisfaction | Audience satisfaction | 20. Audience satisfaction with using CaaSSET modelling and code generation tools to develop a context web service | 0% | 50% | 50% | 0% | 0% | 0% | 100% |
| | | | 21. Audience satisfaction with using CaaSSET API to deliver the context web service appropriately | 0% | 50% | 50% | 0% | 0% | 0% | 100% |

### 7.3.2 Usability survey audience

The survey statistical population was developers in Isfahan and Birjand cities of Iran familiar with context service modelling, development and delivery; we call this *domain knowledge*. The statistical population size was about 53 people. According to the Morgan table method (Krejcie and Morgan, 1970), the sample size should be 32. We used the simple random sampling method without placement, and after communicating with different people in the statistical population, 20 people participated in our survey. Participants self-assessed three dimensions of their *domain knowledge* by answering six questions. Due to the specialized nature of the CaaSSET toolset, the evaluators' *domain knowledge* should not be too low. Therefore, to preserve the survey quality, we eliminated the participants whose *domain knowledge* was less than eight, meaning the lower quartile of *domain knowledge*. Finally, we collected the feedback of 12 developers, who were in the top three quartiles of *domain knowledge*. Among these 12 participants, two had a PhD in Software Engineering, two were third-year PhD students of Software Engineering, four had MSc in Software Engineering or Information Technology, and four were MSc students of Information Technology. We elaborated the results of this self-assessment step in Appendix A (Table A.1).

The evaluators' relevant experience was between two and eight years. Two evaluators had seven or eight years of academic experience, two had six years of relevant experience in both industry and academia, four of them had five years of industrial experience, and four had about two years of academic experience. The evaluators' age was between 25 and 38 years. Two people were 37 to 38 years old, one person was 33 years old, five people were 28 to 29 years old, and the remaining four were 25 to 26 years old.

The evaluators' experience included a combination of knowledge of software design and modelling, model-driven development, UI/UX[5] design, web service development and context-aware application development. Due to the novelty and multidisciplinary subject of the evaluated toolset, the average score of evaluators' *domain knowledge* was 16.2 out of 30, indicating that the developers' average knowledge of the problem domain was mediocre. Besides, the average score of the evaluators' *domain knowledge* ranged

[5] User Interface/User Experience.

from 8 to 25. In other words, a combination of less experienced and more experienced evaluators was used to assess the toolset.

### 7.3.3 Usability evaluation results

After preparing the usability assessment questionnaire, we provided a guideline to develop a context service and deliver it as a web service. We also had a 40-minute online meeting in which we trained 12 participants on how to work with the CaaSSET toolset. Afterwards, the participants modelled and generated a similar context web service. To observe and execute the generated context web service, they used a graphical user interface, which we provided for the CaaSSET API. Finally, we collected the participants' feedback. Participants were allowed to leave the answer blank or choose from five options: *excellent*, *good*, *moderate*, *weak* and *very weak*. We applied the descriptive statistics to calculate for each option the ratio to the total number of received responses. Table 2 shows the usability evaluation results. To make the results more understandable, in the last left column of Table 2, we considered a column called *satisfaction ratio*, showing what percentage of participants had chosen the acceptable answer. The value of this column is derived by adding the values of the columns *excellent* and *good*. We calculated the minimum *satisfaction ratio* for each of the four usability variables by minimizing the values of the *satisfaction ratio* column.

| **Usability question** | **Usability variable** | **Minimum satisfaction ratio** |
|---|---|---|
| Ease of learning and use | Easy to use the CaaSSET toolset | 75% |
| | Easy to learn the CaaSSET toolset | 75% |
| Audience satisfaction and usefulness | The usefulness of the CaaSSET toolset (for facilitating context service development and delivery) | 75% |
| | Audience satisfaction with the CaaSSET toolset | 100% |

***Table 3:*** *Overall results of CaaSSET toolset usability evaluation*

As illustrated in Table 3, the minimum *satisfaction ratio* for the three usability variables is 75%. Overall, more than 75% of the participants evaluated the CaaSSET toolset usability as satisfying (*good* or *excellent*).

In the following, we analyse the descriptive statistics of the usability questionnaire.

- **First evaluation question (easy to learn and use):** More than two-thirds of survey participants rated their modelling knowledge as *moderate*, *weak* or *very weak*. However, 100% of the participants evaluated the ease of working with the CaaSSET toolset as satisfying. More than 75% of the survey participants evaluated five other factors related to ease of use (i.e., user-friendliness, flexibility, consistency and coordination, error-checking and fast recovery) as satisfying. In addition, 91.7% of the participants rated the ease of learning and use of modelling and code generation tools as satisfying, and 75% found it easy to learn how to work with the CaaSSET API. Besides, about 17% of the developers rated the user-friendliness of the CaaSSET modelling tool as *moderate* or *weak*. Hence, improving the visual aspects of the CaaSSET modelling tool can be the subject of our future developments.
- **Second evaluation question (audience satisfaction and usefulness):** 100% of the participants were satisfied with the CaaSSET toolset. Besides, all the participants evaluated the CaaSSET toolset applicability as satisfying. More than 75% of participants also considered the other three usefulness factors (including productivity, time saving, requirement coverage, productivity and effectiveness) satisfying.

### 7.3.4 Threats to validity of usability evaluation

**Number of respondents:** We used a relatively small number of evaluators due to two limitations. The first limitation is the novelty and multidisciplinary subject of the evaluated toolset. The evaluators' experience should be a combination of their knowledge of software design and modelling, model-driven development, UI/UX design, web service development and context-aware application development. Unfortunately, developers with a combination of such expertise are rare, and identifying and communicating with them to participate in a non-profit study was a challenging task. The second limitation was the installation of the prerequisite software (such as Eclipse and its associated tools) before the beginning of the evaluation.

Due to these limitations, our assessment was conducted in two cities where we had previous knowledge of relevant experts, and we had, in the unusual circumstances of COVID-19, physical access to relevant experts for initial coordination, training, software installation and final evaluation. To efficiently increase the number of evaluators, in future research, we are going to work on creating a web-based version of the CaaSSET modelling tool so that more experts can evaluate it remotely without the installation of the Eclipse software and its associated tools.

**Relation of evaluators' domain knowledge and ease of learning and use:** The other threat to our usability survey validity could be that if people with lower domain knowledge had performed the assessment, the toolset usability score would have been lower. In response to this threat, we should note that the participants' average domain knowledge is about 16.2 out of 30, indicating that the evaluators' average *domain knowledge* was *moderate*. The average score of the evaluators' *domain knowledge* also ranged from 8 to 25, indicating that we used a combination of less experienced and more experienced evaluators.

Nevertheless, to fully resolve this threat, we designed six hypotheses, similar to hypothesis H1.1; see Appendix A (Table A.2). These hypotheses examine whether there is a significant positive or negative correlation between the evaluators' *domain knowledge* and the CaaSSET toolset usability.

**Hypothesis H1.1:** There is a significant correlation between evaluators' *domain knowledge* and *ease of learning and use* of the CaaSSET toolset.

We used the Pearson correlation test with a confidence interval level of 95% and a margin error of 5% to prove or disprove the above six hypotheses. The results of analysing these six hypotheses are presented in Appendix A (Table A.2). This analysis showed a weak positive correlation between evaluators' *domain knowledge* and the CaaSSET toolset usability. The correlation is more prominent in the *knowledge of service development and delivery*; however, considering the rejection of the six designed hypotheses, the correlation is not statistically significant and cannot be generalized to the whole statistical population. Hence, according to the usability evaluation results (Table 3) and the correlation analysis (Table A.2), it is concluded that the CaaSSET toolset is usable for audiences above three quarters in the domain knowledge of context service modelling, development and delivery. Therefore, the CaaSSET toolset could fully support ease of context service development and delivery (R13).

## 7.4 Comparison of the proposed solution and related works

This section reviews the most important works related to CASC development and execution. The most important criterion for including a related work in our comparison table is to what extent it supports requirement specifications R1 to R13. We have chosen a combination of related works such that each requirement specification is covered by at least one related work. We can classify works related to the CASC development and execution into four general sections: service composition (SC), CASC, context-aware business processes (CABP) and context service development and delivery (CSD&D).

According to our previous investigations (Moradi et al., 2020), the studies related to CSD&D can be classified into four general categories: context-aware services, context-aware applications, CaaS delivery and CaaS development. In our previous work (Moradi et al., 2020), we compared 16 key approaches, classified in the four mentioned categories, based on 44 evaluation criteria, then elaborated 14 requirements related to CSD&D. In the present research, among the 16 key mentioned approaches, we have chosen five works that are more related to context-aware service composition and execution. Hence, five approaches including CaaSSET (Moradi et al., 2020), ContextUML (Sheng et al., 2010; Sheng and Benatallah, 2005), NGSI (Open Mobile Alliance, 2012), NGSI-LD (ETSI, 2019), and CSDL+CDQL (Hassani et al., 2019) have been included in our new comparison table.

Table 4 compares CA-BPEL (including PCASO, CaaSSET and standard WS-BPEL tools) with 14 related works classified in four categories, based on the 13 requirement specifications R1 to R13. Note that in the CA-BPEL approach, we assumed that standard WS-BPEL-compliant tools support R1 to R2; the PCASO middleware meets R3 to R8, and the CaaSSET framework satisfies R9 to R13.

As shown in Table 4, none of the thirteen related works could fully support R1 to R13. Table 4 also shows that Hagin's approach provides more substantial support for 13 identified requirements and, as such, it has an advantage over most related work. Nevertheless, Hagin's approach and the other investigated related works are weak in supporting R3 to R5 and R12 to R13. In other words, existing approaches either partially support embedding context awareness into a standard composite service (R3 to R5), or have a weakness in considering a comprehensive context model (R12), or they have not provided an easy-to-use solution to facilitate CASC development (R13). Moreover, most existing approaches use a proprietary solution, inconsistent with industry-accepted standards, such as WS-BPEL (R2).

As implied from Table 4 and previously elaborated in our previous work (Moradi et al., 2020), the CaaSSET framework fully supports the development of a comprehensive context service model, whereas the related studies are weak at this (R12). Note that CaaSSET is equipped with an API and a context repository that allows the storage of context values and accessing the context values at run time (R11). Moreover, CaaSSET does not fully support the interoperability between composite services and context web services (R6). However, this weakness is resolved by applying the wrapping adapter of the PCASO middleware to the REST interface of the CaaSSET API. Although CaaSSET fully supports ease of context service development and delivery, it is not designed to facilitate CASC development (R8). Fortunately, this weakness has been resolved by using the standard WS-BPEL development tools.

***Table 4:*** *Comparison of CA-BPEL approach and related work based on R1 to R13*

| | **Evaluation criteria** | **SC** | | **CASC** | | | | **CABP** | | | | **CSD&D** | | | | **CA-BPEL** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Classification** | CASC software requirement specifications (R1 to R13) | UML4SOA (Mayer et al., 2008) | Standard BPEL tools | Baidouri et al. (Baidouri et al. (2012) | Furno and Zimeo (2014) | SABPEL (Cherif et al., 2016) | (Hagin, 2011)+standard BPEL engines | Kocurova (2013) | Wenzl and Strembeck (2013) | Mattos et al. (2014) | Janiesch and Kuhlenkamp (2019) | ContextUML (Sheng and Benatallah, 2005) + ContextServ (Sheng et al., 2010) | NGSI (Open Mobile Alliance, 2012) | NGSI-LD (ETSI, 2019) | CSDL + CDQL (Hassani et al., 2019) | CaaSSET (Moradi et al., 2020) | PCASO + Standard BPEL tools |
| **Service composition and execution** | **R1:** Supporting the development of standard service composition models | ● | ● | ○ | ○ | ◐ | ● | ○ | ◐ | ◐ | ● | ◐ | ○ | ○ | ○ | ○ | ● |
| | **R2:** Compatibility of composite services with standard service composition execution engines | ● | ● | ○ | ○ | ◐ | ● | ○ | ○ | ○ | ● | ◐ | ○ | ○ | ○ | ○ | ● |
| **Embed context awareness into composite services** | **R3:** Push notification context trigger | ○ | ○ | ○ | ○ | ○ | ◐ | ○ | ○ | ○ | ◐ | ◐ | ○ | ○ | ○ | ○ | ● |
| | **R4:** Periodic pull context trigger | ○ | ○ | ○ | ○ | ○ | ◐ | ○ | ○ | ○ | ● | ◐ | ○ | ○ | ○ | ○ | ● |
| | **R5:** Synchronous or asynchronous contextual pre/post-condition for each action of a composite service | ○ | ○ | ○ | ◐ | ◐ | ◐ | ◐ | ◐ | ○ | ◐ | ◐ | ○ | ○ | ○ | ○ | ● |
| | **R6:** Interoperability of composite services with various web service technologies (especially REST and SOAP) | ○ | ○ | ◐ | ○ | ○ | ● | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ● |
| | **R7:** Compatibility of CASC with standard service composition execution engines | ○ | ○ | ◐ | ○ | ◐ | ● | ○ | ○ | ○ | ● | ◐ | ○ | ○ | ○ | ○ | ● |
| | **R8:** Ease of CASC development | ◐ | ◐ | ○ | ◐ | ◐ | ● | ○ | ◐ | ○ | ◐ | ◐ | ○ | ○ | ○ | ○ | ● |
| **Context service development and delivery** | **R9:** Context delivery as a web service, independently of a composite service | ○ | ◐ | ○ | ○ | ◐ | ● | ○ | ○ | ◐ | ◐ | ○ | ● | ● | ◐ | ● | ○ |
| | **R10:** Sharing of the context values with multiple context-aware composite services through a standard API | ○ | ◐ | ○ | ○ | ◐ | ● | ○ | ○ | ● | ◐ | ◐ | ● | ● | ● | ● | ○ |
| | **R11:** Easy run-time access to current and historical context values via a standard API | ○ | ○ | ○ | ○ | ○ | ◐ | ◐ | ○ | ○ | ◐ | ○ | ● | ● | ○ | ● | ○ |
| | **R12:** Support for comprehensive context service model | ○ | ○ | ◐ | ◐ | ◐ | ◐ | ◐ | ○ | ◐ | ◐ | ◐ | ◐ | ◐ | ◐ | ● | ○ |
| | **R13:** Ease of context service development and delivery | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ◐ | ◐ | ◐ | ◐ | ● | ○ |
| | ○ **No support** ◐ **Partial support** ● **Full support** | | | | | | | | | | | | | | | | |

In general, it can be concluded that while related work partially supports requirement specifications R1 to R13, CA-BPEL supports the 13 identified requirement specifications fully.

## 8 Conclusions

The goal of this study was to facilitate the development and execution of context-aware service orchestrations. We achieved this goal by proposing the CA-BPEL approach, which is based on MDD and CaaS. Our main contributions are the CA-BPEL approach and the PCASO middleware, consisting of a reference model and five intermediate web services. PCASO web services include trigger services (a periodic pull service and a push notification service), a listening service, a context evaluation service and a context service adapter. The CA-BPEL approach assumes that instead of modifying the WS-BPEL language meta-model, context is delivered as a web service. Hence, standard composite services can work with a context service similar to other standard web services. The PCASO middleware allows the developer to turn a standard orchestrated service into a context-aware composite service. PCASO is also fully compliant with the WS-BPEL language, accepted by service composition development environments and service composition execution engines. PCASO also facilitates the interoperability of standard composite services and context services.

We showed the usefulness of CA-BPEL by developing and executing a tourism demonstration. We also compared the CA-BPEL with 14 baseline techniques in terms of 13 identified requirements. These evaluations showed that CA-BPEL (i.e., PCASO, CaaSSET and standard WS-BPEL-compliant tools) facilitates CASC development and execution and has advantages over related work. Our investigations also indicated that while CA-BPEL supports the 13 identified requirements, related studies either support limited dimensions of contextual concepts or have a weakness in the critical context awareness features or use a proprietary solution, inconsistent with industry-accepted standards.

The judgment on the convenience of CASC development and execution is entirely dependent on satisfying requirement R13, i.e., ease of context service development and delivery. As such, we conducted a usability survey to measure the satisfaction of R13 by the CaaSSET toolset. In the survey, we asked 12 developers to evaluate four usability variables: ease of use, ease of learning, usefulness and audience satisfaction. The survey results showed that 100% of the participants were satisfied with the CaaSSET toolset for facilitating context service development and delivery as a web service. All the participants evaluated the applicability and ease of working with the CaaSSET toolset as satisfying. Overall, more than 75% of the survey participants ranked the CaaSSET toolset usability as good or excellent. To resolve threats to the survey validity, we used a Pearson correlation analysis, showing that the CaaSSET toolset is usable for audiences above three-quarters in the knowledge of context service modelling, development and delivery. The significance of this study is in demonstrating that MDD, along with CaaS, has much potential to facilitate the development of context-aware service compositions.

This research can be extended in several ways. For instance, we can work on a web-based version of the CaaSSET toolset so that its audience would not need to install Eclipse IDE and its associated tools. We can also explore a brand new solution to facilitate context-aware service choreography development. CA-BPEL can also be customized for embedding context awareness into service choreography standards, the BPMN standard or non-standard commercial service composition platforms, such as Amazon AWS Step Functions and Netflix Conductor. According to our preliminary investigations, such customizations can be made with minor modifications to the CaaSSET framework, which is the core of the CA-BPEL approach.

## Additional Information and Declarations

**Conflict of Interests:** The authors declare no conflict of interest.

**Author Contributions:** H.M.: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Writing – Original Draft, Writing – Review & Editing, Visualization. B.Z.: Project administration, Supervision, Methodology, Writing – Review & Editing, Resources, Validation. K.Z.: Writing – Review & Editing, Validation.

**Data Availability:** The data that support the findings of this study are available from the corresponding author upon reasonable request.

## Appendix A

***Table A.1:*** *CaaSSET toolset evaluators' domain knowledge*

| Domain knowledge | Knowledge question | Excellent (5) | Good (4) | Moderate (3) | Weak (2) | Very weak (1) |
|---|---|---|---|---|---|---|
| Modelling knowledge | Mastering modelling standards and modelling tools | 16.7% | 25% | 16.7% | 25% | 16.7% |
| | Mastering the MDD approach | 25% | 0% | 0% | 41.7% | 33.3% |
| Pervasive computing knowledge | Mastering the development of context-aware applications | 8.3% | 8.3% | 33.3% | 33.3% | 16.7% |
| | Mastering the development of context services | 8.3% | 8.3% | 8.3% | 33.3% | 41.7% |
| Knowledge of service development and delivery | Mastering the development of web services | 8.3% | 25% | 33.3% | 8.3% | 25% |
| | Mastering the delivery of Software as a Service (SaaS) | 16.7% | 8.3% | 41.7% | 25% | 8.3% |

***Table A.2:*** *Correlation analysis between evaluators' domain knowledge and CaaSSET toolset usability*

| Hypothesis | Domain knowledge | Toolset usability | Number of instances | P-value | Significance level | Correlation type | Correlation result |
|---|---|---|---|---|---|---|---|
| H1.1 | Modelling knowledge | Ease of learning and use | 12 | 0.203 | 0.528 | Positive (very weak) | Rejected |
| H1.2 | Pervasive computing knowledge | Ease of learning and use | 12 | 0.111 | 0.732 | Positive (very weak) | Rejected |
| H1.3 | Knowledge of service development and delivery | Ease of learning and use | 12 | 0.377 | 0.227 | Positive (weak) | Rejected |
| H2.1 | Modelling knowledge | Audience satisfaction and usefulness | 12 | 0.117 | 0.718 | Positive (very weak) | Rejected |
| H2.2 | Pervasive computing knowledge | Audience satisfaction and usefulness | 12 | 0.118 | 0.715 | Positive (very weak) | Rejected |
| H2.3 | Knowledge of service development and delivery | Audience satisfaction and usefulness | 12 | 0.407 | 0.189 | Positive (weak) | Rejected |

## References

**Abrahão, S., Insfran, E., Sluÿters, A., & Vanderdonckt, J.** (2021). Model-based intelligent user interface adaptation: Challenges and future directions. *Software and Systems Modeling*, *20*(5), 1335–1349. https://doi.org/10.1007/s10270-021-00909-7

**Armstrong, C.** (2014). *Understanding Reference Models and Reference Architectures*. SATURN. https://resources.sei.cmu.edu/asset_files/Presentation/2014_017_101_90458.pdf

**Baidouri, H., Hafiddi, H., Nassar, M., & Kriouile, A.** (2012). Towards a context-aware composition of services. *International Journal of Computer Science and Network Security*, *12*(3), 133–140.

**Baldauf, M., Dustdar, S., & Rosenberg, F.** (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, *2*(4), 263. https://doi.org/10.1504/IJAHUC.2007.014070

**Bézivin, J.** (2004). In search of a basic principle for model driven engineering. *Novatica Journal*, *5*(2), 21–24.

**Botangen, K. A., Yu, J., Han, Y., Sheng, Q. Z., & Han, J.** (2020). Quantifying the adaptability of workflow-based service compositions. *Future Generation Computer Systems*, *102*, 95–111. https://doi.org/10.1016/j.future.2019.08.010

**Brambilla, M., Cabot, J., & Wimmer, M.** (2012). *Model-Driven Software Engineering in Practice*. Morgan & Claypool Publishers.

**Chabridon, S., Conan, D., Abid, Z., & Taconet, C.** (2013). Building ubiquitous QoC-aware applications through model-driven software engineering. *Science of Computer Programming*, *78*(10), 1912–1929. https://doi.org/10.1016/j.scico.2012.07.019

**Cherif, S., Ben Djemaa, R. B., & Amous, I.** (2016). A user-aware approach for describing and publishing context aware composite Web service. *International Journal of Pervasive Computing and Communications*, *12*(2), 174–193. https://doi.org/10.1108/ijpcc-01-2016-0011

**ETSI.** (2019). *Context Information Management (CIM); NGSI-LD API*. https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.01.01_60/gs_CIM009v010101p.pdf

**Faieq, S., Saidi, R., El Ghazi, H., Front, A., & Rahmani, M. D.** (2021). Building adaptive context-aware service-based smart systems. *Service Oriented Computing and Applications*, *15*(1), 21–42. https://doi.org/10.1007/s11761-020-00310-0

**Furno, A., & Zimeo, E.** (2014). Context-aware Composition of Semantic Web Services. *Mobile Networks and Applications*, *19*(2), 235–248. https://doi.org/10.1007/s11036-014-0494-y

**Gartner.** (2019). *Magic Quadrant for Intelligent Business Process Management Suites*. http://dba.nextblue.ca/wp-content/uploads/sites/3/2019/07/Magic-Quadrant-for-Intelligent-Business-Process-Management-Suites.pdf

**Gartner.** (2022). *Gartner Top Strategic Technology Trends for 2022*. https://www.gartner.com/en/information-technology/insights/top-technology-trends

**Hagin, R.** (2011). *Enabling integration and aggregation of context information into WS-BPEL processes.* Thesis. University of Stuttgart.

**Hassani, A., Medvedev, A., Delir Haghighi, P., Ling, S., Zaslavsky, A., & Prakash Jayaraman, P.** (2019). Context Definition and Query Language: Conceptual Specification, Implementation, and Evaluation. *Sensors*, *19*(6), Article no. 1478. https://doi.org/10.3390/s19061478

**Hassani, A., Medvedev, A., Haghighi, P. D., Ling, S., Indrawan-Santiago, M., Zaslavsky, A., & Jayaraman, P. P.** (2018). Context-as-a-Service Platform: Exchange and Share Context in an IoT Ecosystem. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, (pp. 385–390). IEEE. https://doi.org/10.1109/PERCOMW.2018.8480240

**Hoyos, J. R., García-Molina, J., Botía, J. A., & Preuveneers, D.** (2016). A model-driven approach for quality of context in pervasive systems. *Computers & Electrical Engineering*, *55*, 39–58. https://doi.org/10.1016/j.compeleceng.2016.07.002

**Hynes, G., Reynolds, V., & Hauswirth, M.** (2009). A Context Lifecycle for Web-Based Context Management Services. In P. Barnaghi, K. Moessner, M. Presser, & S. Meissner (Eds.), *Smart Sensing and Context* (pp. 51–65). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04471-7_5

**Ibrahim, N. I.** (2012). *Specification, composition and provision of trustworthy context-dependent services.* Dissertation. Concordia University.

**Janiesch, C., & Kuhlenkamp, J.** (2019). Enhancing business process execution with a context engine. *Business Process Management Journal*, *25*(6), 1273–1290. https://doi.org/10.1108/BPMJ-06-2017-0160

**Kerpedzhiev, G. D., König, U. M., Röglinger, M., & Rosemann, M.** (2021). An Exploration into Future Business Process Management Capabilities in View of Digitalization: Results from a Delphi Study. *Business & Information Systems Engineering*, *63*(2), 83–96. https://doi.org/10.1007/s12599-020-00637-0

**Kocurova, A.** (2013). *Distributed collaborative context-aware content-centric workflow management for mobile devices.* Dissertation. University of West London.

**Krejcie, R. V, & Morgan, D. W.** (1970). Determining sample size for research activities. *Educational and Psychological Measurement*, *30*(3), 607–610.

**Laleh, T.** (2018). *Context Verification and Adaptation in Web Service Composition.* Dissertation. Concordia University.

**Lübke, D., & Pautasso, C.** (Eds.). (2019). *Empirical Studies on the Development of Executable Business Processes*. Springer International Publishing. https://doi.org/10.1007/978-3-030-17666-2

**Lund, A. M.** (2001). Measuring usability with the use questionnaire. *Usability Interface*, *8*(2), 3–6.

**Mattos, T. da C., Santoro, F. M., Revoredo, K., & Nunes, V. T.** (2014). A formal representation for context-aware business processes. *Computers in Industry*, *65*(8), 1193–1214. https://doi.org/10.1016/j.compind.2014.07.005

**Mayer, P., Schroeder, A., & Koch, N.** (2008). MDD4SOA: Model-Driven Service Orchestration. In *12th International IEEE Enterprise Distributed Object Computing Conference*, (pp. 203–212). IEEE. https://doi.org/10.1109/EDOC.2008.55

**Moore, P., Xhafa, F., & Barolli, L.** (2014). Context-as-a-Service: A Service Model for Cloud-Based Systems. In *2014 Eighth International Conference on Complex, Intelligent and Software Intensive Systems*, (pp. 379–385). IEEE. https://doi.org/10.1109/CISIS.2014.53

**Moradi, H., Zamani, B., & Zamanifar, K.** (2020). CaaSSET: A Framework for Model-Driven Development of Context as a Service. *Future Generation Computer Systems*, *105*, 61–95. https://doi.org/10.1016/j.future.2019.11.028

**Nikoo, M. S., Babur, Ö., & van den Brand, M.** (2020). A survey on service composition languages. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, (pp. 1–5). ACM. https://doi.org/10.1145/3417990.3421402

**Open Mobile Alliance.** (2012). *NGSI Context Management*. http://www.openmobilealliance.org/release/NGSI/V1_0-20120529-A/OMA-TS-NGSI_Context_Management-V1_0-20120529-A.pdf

**Papazoglou, M. P., & van den Heuvel, W.-J.** (2007). Service oriented architectures: Approaches, technologies and research issues. *The VLDB Journal*, *16*(3), 389–415. https://doi.org/10.1007/s00778-007-0044-3

**Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S.** (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77.

**Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D.** (2014). Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, *16*(1), 414–454. https://doi.org/10.1109/SURV.2013.042313.00197

**Salber, D., Dey, A. K., & Abowd, G. D.** (1999). The Context Toolkit: Aiding the Development of Context-Aware Applications. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 434–441). ACM. https://doi.org/10.1145/302979.303126

**Schefer-Wenzl, S., & Strembeck, M.** (2013). Modelling context-aware RBAC models for mobile business processes. *International Journal of Wireless and Mobile Computing*, *6*(5), 448–462. https://doi.org/10.1504/IJWMC.2013.057387

**Sheng, Q. Z., & Benatallah, B.** (2005). ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services Development. In *International Conference on Mobile Business*, (pp. 206–212). IEEE. https://doi.org/10.1109/ICMB.2005.33

**Sheng, Q. Z., Qiao, X., Vasilakos, A. V, Szabo, C., Bourne, S., & Xu, X.** (2014). Web services composition: A decade's overview. *Information Sciences*, *280*, 218–238. https://doi.org/10.1016/j.ins.2014.04.054

**Sheng, Q. Z., Yu, J., Segev, A., & Liao, K.** (2010). Techniques on developing context-aware web services. *International Journal of Web Information Systems*, *6*(3), 185–202. https://doi.org/10.1108/17440081011070141

**Vahdat-Nejad, H.** (2014). Context-aware middleware: A review. In Brézillon P., Gonzalez A. (eds.), *Context in computing* (pp. 83–96). Springer. https://doi.org/10.1007/978-1-4939-1887-4_6

**Vom Brocke, J., Baier, M.-S., Schmiedel, T., Stelzl, K., Röglinger, M., & Wehking, C**. (2021). Context-Aware Business Process Management: Method Assessment and Selection. *Business & Information Systems Engineering*, *63*(5), 533–550. https://doi.org/10.1007/s12599-021-00685-0

**Wenzl, S. S., & Strembeck, M.** (2013). Modelling context-aware RBAC models for mobile business processes. *International Journal of Wireless and Mobile Computing*, *6*(5), 448–462. https://doi.org/10.1504/IJWMC.2013.057387

**Yigitbas, E., Jovanovikj, I., Biermeier, K., Sauer, S., & Engels, G.** (2020). Integrated model-driven development of self-adaptive user interfaces. *Software and Systems Modeling*, *19*(5), 1057–1081. https://doi.org/10.1007/s10270-020-00777-7

**Zhao, X., Yongchareon, S., & Cho, N.-W.** (2021). Enabling situational awareness of business processes. *Business Process Management Journal*, *27*(3), 779–795. https://doi.org/10.1108/BPMJ-07-2020-0331

**Zhou, J., Gilman, E., Palola, J., Riekki, J., Ylianttila, M., & Sun, J.** (2011). Context-aware pervasive service composition and its implementation. *Personal and Ubiquitous Computing*, *15*(3), 291–303. https://doi.org/10.1007/s00779-010-0333-5

**Editorial record:** The article has been peer-reviewed. First submission received on 10 November 2021. Revisions received on 6 January 2022. Accepted for publication on 18 January 2022. The editor in charge of coordinating the peer-review of this manuscript and approving it for publication was Zdenek Smutny.

Acta Informatica Pragensia is published by Prague University of Economics and Business, Czech Republic.

ISSN: 1805-4951