Designing and Building Context-Aware Services: The ContextServ Project

Quan Z. Sheng[1], Jian Yu[2], Wei Emma Zhang[3], Shuang Wang[1,4], Xiaoping Li[4], and Boualem Benatallah[5]

[1]Department of Computing, Macquarie University, NSW 2109, Australia
[2]Department of Computer Science, Auckland University of Technology, New Zealand
[3]School of Computer Science, the University of Adelaide, SA 5005, Australia
[4]School of Computer Science and Engineering, Southeast University, Nanjing, China
[5]School of Computer Science and Engineering, UNSW, NSW 2052, Australia

Abstract. In the era of Web of Things and services, context-aware services (CASs) are emerging as an important technology for building innovative smart applications. CASs enable the information integration from both the physical and virtual world, which affects the way human live. However, it is still challenging to build CASs, due to lack of context provisioning management approach and lack of generic approach for formalizing the development process. In this paper, we briefly introduce a large research project, ContextServ, which provides a platform for model-driven development of CASs based on a UML-based modelling language. We discuss the literature and also highlight several future research opportunities for context-aware service research and development.

Keywords: Context-aware services, Internet of Things, model driven development, modeling language, ContextUML, adaptive services

1 Introduction

Over the years, the Web has gone through many transformations, from traditional linking and sharing of computers and documents (i.e., "Web of Data") to current connecting of people (i.e., "Web of People"). With the recent advances in radio-frequency identification technology, sensor networks, and Web services, the Web is continuing the transformation and will be slowly evolving into the so-called "Web of Things and Services" [13, 27]. Indeed, this future Web will provide an environment where everyday physical objects such as buildings, sidewalks, and commodities are readable, recognizable, addressable, and even controllable using services via the Web. The capability of integrating the information from both the physical world and the virtual one not only affects the way how we live, but also creates tremendous new Web-based business opportunities such as support of independent living of elderly persons, intelligent traffic management, efficient supply chains, and improved environmental monitoring [23, 27]. Therefore, context awareness, which refers to the capability of an application or a service being aware of its physical environment or situation (i.e.,

context) and responding proactively and intelligently based on such awareness [1, 15, 19], has been identified as one of the key challenges and most important trends in computing today and holds the potential to make our daily lives more productive, convenient, and enjoyable.

Nowadays, Web services have become a major technology to implement loosely coupled business processes and perform application integration [24, 36]. Through the use of context, a new generation of smart Web services is currently emerging as an important technology for building innovative context-aware applications. We call such category of Web services as context-aware Web services (CASs). CASs are emerging as an important technology to underpin the development of new applications (user centric, highly personalized) on the future ubiquitous Web. A CAS is a service that uses context information to provide relevant information and/or services to users [19, 31, 9, 6]. A CAS can present relevant information or can be executed or adapted automatically, based on available context information.

Although the combination of context awareness and Web services sounds appealing, injecting context into services raises a number of significant challenges, which have not been widely recognized or addressed by the services community [31, 36, 30]. One reason for this difficulty is that current Web services standards, such as the Web Services Description Language (WSDL), Web Application Description Language (WADL), and the Simple Object Access Protocol (SOAP), are not sufficient for describing and handling context information. CAS developers must implement everything related to context management, including collection, dissemination, and usage of context information, in an ad hoc manner. Another reason is that, CASs are frequently required to be dynamically adaptive in order to cope with constant changes, which means a service being able to change its behavior at runtime in accordance with the contexts. Unfortunately, service-oriented systems built with WS-BPEL (Web Services Business Process Execution Language) are still too rigid. The third reason is, to the best of our knowledge, there is a lack of generic approaches for formalizing the development of CASs. As a consequence, developing and maintaining CASs is a very cumbersome, error-prone, and time consuming activity, especially when these CASs are complex.

In this paper, we will first give an overview of the ContextServ project, which provides a comprehensive platform that supports the full lifecycle of CASs development, including a visual ContextUML editor, a ContextUML to WS-BPEL translator, and a WS-BPEL deployer (see Figure 1). Another feature of ContextServ is that it supports dynamic adaptation of WS-BPEL based context-aware composite services by weaving context-aware rules into the process. ContextServ exploits a model-driven approach that offers significant design flexibility by separating the context modeling and context awareness from service components, which eases both development and maintenance of CASs. It also supplies a set of automated tools for generating and deploying executable implementations of CASs. We will then review the relevant literature and highlight several future research opportunities for CAS research and development.

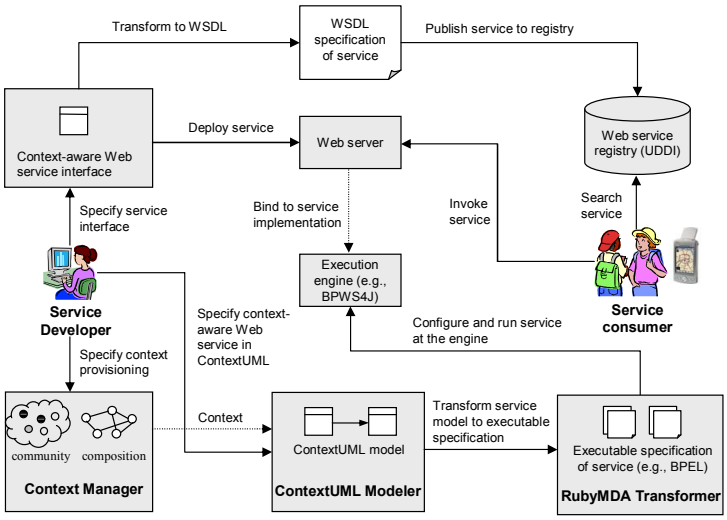

Fig. 1. Architecture of the ContextServ platform

2 The ContextServ Project

ContextServ adopts model-driven development (MDD). The basic idea of MDD is that by adopting a high-level of abstraction, software systems can be specified in platform independent models (PIMs), which are then semi-automatically transformed into platform specific models (PSMs) of target executable platforms using some transformation tools. The same PIM can be transformed into different executable platforms (i.e., multiple PSMs), thus considerably simplifying software development. This section will briefly introduce the ContextUML language, the RubyMDA transformer, and the adaptive CAS process.

2.1 The ContextUML Language

ContextServ relies on ContextUML [28], a UML-based modeling language that provides high-level, visual constructs for specifying context-aware Web services. As shown in Figure 2, ContextUML metamodel consists of three main parts: the *context modeling metamodel*, the *context-awareness modeling metamodel*, and the *service modeling metamodel*. We will focus on introducing the first two parts since the service modeling metamodel (the left part of Figure 2) follows the standard service definitions.

Context Modeling. In ContextUML, a context is further distinguished into two categories that are formalized by *AtomicContext* and *CompositeContext*. Atomic contexts are low-level contexts that do not rely on other contexts and can be provided directly by context sources. In contrast, composite contexts are high-level contexts that may not have direct counterparts on the context provision. A

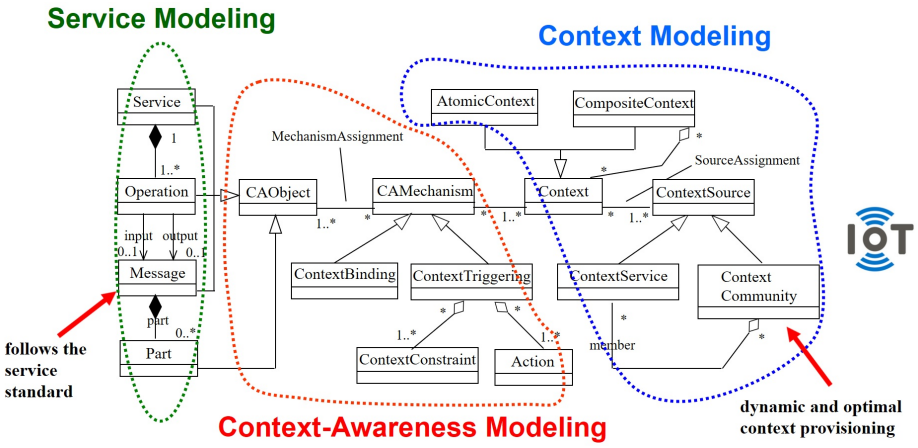

Fig. 2. The ContextUML metamodel in ContextServ

composite context aggregates multiple contexts, either atomic or composite. The concept of composite context can be used to provide a rich modeling vocabulary. ContextUML abstracts two categories of context sources, formalized by the context source subtypes *ContextService* and *ContextServiceCommunity*. A context service is provided by an autonomous organization (i.e., context provider), collecting, refining, and disseminating context information.

To solve the challenges of heterogeneous and dynamic context information, we abstract the concept of context service community, which enables the dynamic provisioning of optimal contexts. A context service community aggregates multiple context services, offering with a unified interface. It is intended as a means to support the dynamic retrieval of context information. A community describes the capabilities of a desired service (e.g., providing user's location) without referring to any actual context service (e.g., `WhereAmI` service). When the operation of a community is invoked, the community is responsible for selecting the most appropriate context service that will provide the requested context information. Context services can join and leave communities at any time. By abstracting context service community as one of the context sources, we can enable the dynamic context provisioning. In other words, CAS designers do not have to specify which context services are needed for context information retrieval at the design stage. The decision of which specific context service should be selected for the provisioning of a context is postponed until the invocation of CASs.

The quality of context is extremely important for CASs in the sense that context information is used to automatically adapt services or content they provide. The imperfection of context information may make CASs *misguide* their users. For example, if the weather information is outdated, our attractions searching service might suggest users to surf at the Bondi Beach although it is rainy and stormy. Via context service communities, the optimal context information is always selected, which in turn, ensures the quality of CASs.

Context Awareness Modeling. ContextUML abstracts two context awareness mechanisms, namely *context binding* and *context triggering*. The former models automatic contextual configuration (e.g., automatic invocation of Web services by mapping a context onto a particular service input parameter), with the semantics of that the value of the object is supplied by the value of the context. The context triggering models the situation of contextual adaptation where services can be automatically executed or modified based on context information. A context triggering mechanism contains two parts: a set of *context constraints* and a set of *actions*, with the semantics of that the actions must be executed if and only if all the context constraints are evaluated to true.

Context awareness mechanisms are assigned to context-aware objects, modelled as *CAObject*, by the relation *MechanismAssignment*, indicating which objects have what kinds of context awareness mechanisms. CAObject is a base class of all model elements in ContextUML that represent context-aware objects. There are four subtypes of CAObject: *Service*, *Operation*, *Message*, and *Part*. It should be noted that the four primitives are directly adopted from WSDL, which enables designers to build CASs on top of the previous implementation of Web services.

2.2 ContextUML Modeler and RubyMDA Transformer

In the ContextServ platform, the ContextUML modeler provides a visual interface for defining context-aware Web services using ContextUML [29]. In particular, we extend ArgoUML[1], an existing UML editing tool, by developing a new diagram type, ContextUML diagram, which implements all the abstract syntax of the ContextUML language.

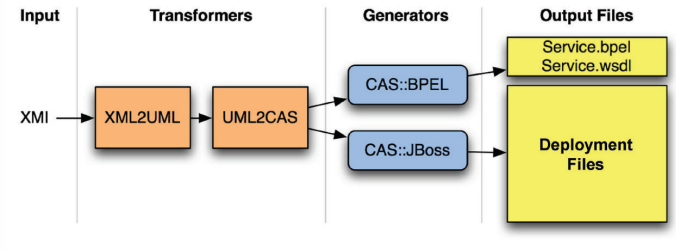

Fig. 3. RubyMDA data flow

Services represented in ContextUML diagrams are exported as XMI files for subsequent processing by the RubyMDA transformer, which is responsible for transforming ContextUML diagrams into executable Web services, using RubyGems[2]. The ContextServ platform currently supports WS-BPEL, a de facto

[1] http://argouml.tigris.org.
[2] https://rubygems.org/.

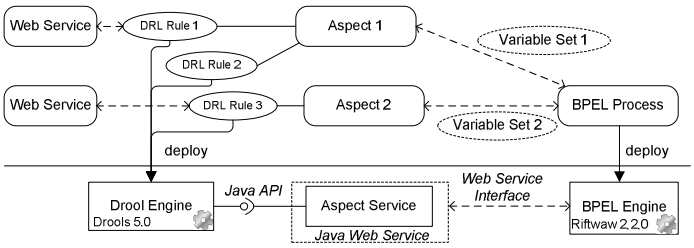

Fig. 4. An anatomy of the adaptive runtime environment

standard for specifying executable processes. Once the BPEL specification is generated, the model transformer deploys the BPEL process to an application server and exposes it as a Web service. In the implementation, JBoss Application Server is used since it is open source and includes a BPEL execution engine jBPM-BPEL. RubyMDA is developed based on the model transformation rules. The model transformation rules are mappings from ContextUML stereotypes to BPEL elements.

Figure 3 shows the data flow of RubyMDA model transformer. RubyMDA takes the XMI document as an input which represents the ContextUML diagram. RubyMDA reads the XMI document and constructs the UML model which is a set of data structure representing the components in UML class diagram. After the UML model is constructed, RubyMDA transforms it into CAS model which is a set of data structure representing the CAS described in ContextUML diagram. Finally, RubyMDA generates a BPEL process and WSDL document for a CAS. Moreover, it generates a set of deployment files needed to deploy CAS to a server.

2.3 Adaptive CAS Processes

As stated by Papazoglou in [24], "*services and processes should equip themselves with adaptive service capabilities so that they can continually morph themselves to respond to environmental demands and changes without compromising operational and financial efficiencies*". It is particularly important to CASs to cope with the functional changes raised from both business requirements and environmental contexts, and bringing dynamic adaptability (or agility) to service processes, which means a CAS should have the ability of behavior adaptation.

We further develop MoDAR PIMs [34, 35] which include the *base model*, the *variable model*, and the *weave model*. The base model represents the relatively stable processing procedures, or *flow* logic, of a CAS system; while the variable model represents the more volatile *decision* aspect of the business requirements. To make the base model and the variable model semantically inter-operable, we use a minimum set of ontology concepts as the basic elements in defining activity parameters in processes and also in defining rule entities. We also adopt an aspect-oriented approach to integrate the base model and the variable model

using a *weave model*. This approach ensures the modularity of the base model and the variable model so that they can evolve independently.

The variable model is automatically transformed into Drools rules, and the weave model is automatically transformed into an abstract BPEL process, where at every *join point*, the invocation to a rule aspect is translated to a special Web service invocation. After the designer manually associates concrete Web services with abstract services in the process to implement their functionalities, the process is automatically transformed into an executable BPEL process. As shown in Fig. 4, the BPEL process and the Drools rules are deployed to their corresponding engines. Dynamic adaptability is achieved in a way that we can freely add/remove/replace business rules defined in the modeling phase and then transform and redeploy them without terminating the execution of the process.

3 Literature Discussions

With the maturing and wide-adopting of Web service technology, research on providing engineering approaches to facilitate the development of context-aware services has gained significant momentum. Using model-driven paradigm to develop CAS has been proven to be a valuable and important strand in this research area considering the quality and efficiency it brings along.

In general, the approaches for developing CASs fall into five categories: i) *Middleware solutions and dedicated service platforms*, ii) *Use of ontologies*, iii) *Rule-based reasoning*, iv) *Source code level programming/Language extensions*, and v) *Message interception* [21]. Each kind of approach has its own pros and cons. For example, the source code level approach can give more freedom to developers to do all kinds of context-aware adaptation, but it does not separate apart the concerns on context-awareness and suffers from a significant maintenance cost. As for the model-driven approach, apart from its advantages, it requires to keep the consistency between high level models and low level executable code at all times, which brings extra complexity.

In this section, we overview the representative research efforts in the literature on model-driven development of context-aware services. Table 1 gives a detailed summary of some representative related works and comparison from the perspectives of modeling language, MDD techniques, tools and platform.

In [5], Ayed et al. proposed a UML metamodel that supports context-aware adaptation of service design from structural, architectural and behavioral perspectives. The structural adaptation can extend the service objects structure by adding or deleting its methods and attributes. The architectural adaptation can add and delete service objects of an application according to the context. The behavioral adaptation can adapt the behavior of the service object by extending its UML sequence diagram with optional context related sequences. Furthermore, based on the UML metamodel, Ayed ed al. proposed an MDD approach to model context-aware applications independently from the platform, which includes six phases that approach step by step the mechanisms required to acquire context information and perform adaptations.

Table 1. Summary and comparison of model-driven approaches for context-aware application development

		Ayed 2008	Sindico-Grassi 2009	Prezerakos 2007	Kapitsaki 2009	Hoyos 2013,2016	Boudaa 2017	ContextServ 2009, 2015
Context modeling	Modeling language (based-Model)	UML	UML (CAMEL)	UML (ContextUML*)	UML (ContextUML*)	DSL (MLContext)	Ontology& UML	UML (ContextUML)
	Atomic context	+	+	+	+	+	+	+
	Composite context	-	+	+	+	+	+	+
	Context quality	+	-	-	-	+	-	+
	Context sensing	+	-	-	-	+	+	-
Service modeling		-	-	+	+	-	-	+
Context-awareness modeling	Context binding	+	+	+	+	+	+	+
	Context triggering	+	+	+	+	+	+	+
	Behavior adaptation	+	-	-	-	-	+	+
Decoupling business logic and context logic		+	+	+	+	+	+	+
Adaptation time (design-time/run-time)		design-time	design-time run-time	design-time run-time	design-time	design-time	design-time run-time	design-time run-time
Implementation platform		unspecified	AspectJ	SOA	SOA	OCP/JCAF	SOA (FraSCAti)	SOA (BPEL)
Supporting software tools	Graphical modeling environment	-	+	+	-	+	+	+
	Transformation tool	-	-	+	+	+	+	+

In [32], Sindico and Grassi proposed CAMEL (Context Awareness ModEling Language) which considers both model-driven development and aspect-oriented design paradigms so that the design of the application core can be decoupled from the design of the adaptation logic. In particular, CAMEL categorizes context into *state-based* which characterizes the current situation of an entity and *event-based* which represents changes in an entitys state. Accordingly, state constraints, which are defined by logical predicates on the value of the attributes of a state-based context, and event constraints, which are defined as patterns of event [7], are used to specify context-aware adaptation feature of the application.

In [17], Hoyos et al. proposed a textual Domain-Specific Language (DSL), namely MLContext, which is specially tailored for modeling context information. It has been implemented by applying MDD techniques to automatically generate software artifacts from context models. The MLContext abstract syntax has been defined as a metamodel, and model-to-text transformations have been written to generate the desired software artifacts (e.g., OCP middleware and JCAF middleware). The concrete syntax has been defined with the EMFText tool, which generates an editor and model injector. Furthermore, in [18], MLContext is extended for modeling quality of context (QoC) and the models can be mapped to code for two frameworks (COSMOS and SAMURA) supporting QoC.

In [25], Prezerakos et al. addressed the decoupling of core service logic from context-related functionality by adopting a model-driven approach based on a modified version of ContextUML [28]. Core service logic and context handling are treated as separate concerns at the model level as well as in the resulting source code. In the design phase, besides class diagrams, UML activity diagrams are used for modeling the core service logic flow in conjunction with MDE (Model-driven Engineering) transformation techniques and AOP (Aspect Oriented Programming). In the coding phase, AOP encapsulates context-dependent behaviors in discrete AspectJ code modules. Context binding information provided in UML models is used to create pointcuts and related advices, as well as to create the binding between them. In [20], Kapitsaki et al. proposed an architecture

for the context adaptation of Web applications consisting of Web services and a model-driven methodology for the development of such context-aware composite applications. In the methodology, the Web application functionality is completely separated from the context adaptation at all development phases (analysis, design and implementation). In the modeling level, composite web applications are modeled in UML and the application design is kept, at a great extent, independent from specific platform implementations and flexible enough to allow the introduction of different code specific mappings. Context adaptation is performed on a service interface level to keep client independent. The modeling exploits a number of pre-defined profiles, whereas the target implementation is based on an architecture that performs context adaptation of web services based on interception of Simple Object Access Protocol (SOAP) messages.

In [12], Boudaa et al. proposed an approach taking advantage of combining MDD and AOP to sustain the development of context-aware service-based applications in mobile and ubiquitous environments. Contexts are modeled with a proposed ontology-based context model which is structured on three sub-ontologies: generic, domain and application ontologies. A UML-based metamodel, called ContextAspect, is proposed to define and specify where and how the context-aware adaptation takes place. The ContextAspect metamodel is composed of three parts: aspect modeling, context modeling and context-awareness modeling. AOM handles the context-awareness logic in ContextAspect models (as variants) to fill context-aware application elements (as variation points) by using weaving techniques at design and run times. At design-time, the weaving enables to produce a wide range of context-aware application models without designing them from the beginning. The run-time weaving consists of weaving necessary reconfiguration into the running application according to the context change, so accomplishing its dynamic adaptation.

To the perspective of modeling language for context-aware application development, we compare ContextUML with the other metamodels from the issues of context modeling, servcie modeling, and context-awareness modeling. It should be noted that, in [25, 20], their models are modified versions of ContextUML, so most of the language capabilities of their models equal to ContextUML's and the comparison with them will not discussed below. As we can see from the table, all languages support the modeling of atomic context. For composite context, although CAMEL claims that atomic contexts can be aggregated but no details were given in the paper. In [12], composite context is inferred from low-level contexts using Semantic Web Rule Language (SWRL), and in MLContext, simple references are used to link composite context with their atomic contexts. ContextUML gives a complete approach to composing a composite context from atomic contexts in statechart which is a widely used formalism integrated into UML. ContextUML supports context quality modeling and use context service community to support QoC-based context selection. Although Ayed UML is able to specify the quality attributes of a context, no runtime support was reported in the paper. In [18], MLContext was extended for modeling QoC. However, it does not support QoC-based context selection.

As to service modeling, only ContextUML directly supports the structure of Web services, which is of enormous importance to the development of context-aware Web services. The other languages just use plain UML classes to represent Web services or even without support of Web services.

For context-awareness modeling, all the languages except MLContext support the main features including context binding and context triggering. However, only ContextUML, Ayed UML and ContextAspect model support behavior adaptation, which means a service or process has the ability to change its behavior at runtime in accordance with the changes in the requirements and/or the external environment(contexts). Ayed UML only supports to define behavior adaptation in design-time, and no run-time support is reported in the paper. Both of ContextServ and ContextAspect model support behavior adaptation in design-time and run-time. The mechanism of behavior adaptation in ContextAspect model enables to change alternatively the application behavior by selecting one among several behaviours in accordance with current contextual situation.

Because dynamic adaptation is closely related to the targeting system, we also listed the supported targeting implementation platform of each approach. Different implementation languages or underlying frameworks/platforms and middleware are adopted in each approach. ContextServ and approaches presented in [25, 20, 12] support the SOA paradigm. After modeling adaptation in ContextUML, it can be transformed and the behavior adaptation will be reflected in standard BPEL that has become a de facto industry standard (widely adopted by major IT service providers including IBM, Oracle, and SAP) to create composite service processes and applications. [12] uses FraSCAti platform as the target platform which supports Service Component Architecture (SCA). Models of MLContext can be transformed to specific context middleware (e.g., OCP and JCAF). CAMEL is still an ongoing work, so only examples on how to transform to ContextJ [16] were described in the paper. As to Ayed UML, no targeting systems are reported in the paper.

For supporting software tools, ContextUML has a comprehensive graphical modeling environment developed on top of ArgoUML and also a full-fledged automatic transformation tool for generating deployable BPEL code. All of CAMEL, MLContext and language in [12] have a graphical modeling environment based on Eclipse EMF[3], but no fully workable transformation tools are reported.

4 Open Research Issues

Although context-aware services have been an active research topic for more than a decade, existing research efforts generally focus more on addressing some specific aspects and lack of a holistic view on the problem [24, 31, 13]. Moreover, the rapid rise and adoption of new computing paradigms such as the Internet

[3] http://www.eclipse.org/modeling/emf/

of Things (IoT), Edge Computing also present compounded challenges in CAS development. In this section, we identify several important directions for future research in this area.

Contextual Data Management. Contextual information is a critical integral component of CASs. There is an urgent need for a holistic approach on the life cycle of contextual data management, from data acquisition, contextual data modeling, reasoning, and transformation, to dissemination. IoT is increasingly becoming an important source for rich and real-time contextual information for CASs. However, the diverse, heterogenous, large scale, and unreliable IoT sensors present significant challenges [27, 26]. This calls for more research on solutions that can effectively aggregate and distill heterogeneous and large IoT data to obtain contextual data of appropriate quality. Future IoT is expected to be 50 to 100 times bigger than the current Internet. This poses a new set of challenges to discover the right IoT devices at the right time and right place for a particular contextual information offering [33, 4]. One technical direction towards IoT discovery is to exploit the textual descriptions associated with IoT devices and perform the *natural order ranking* of IoT contents.

Context-Aware Requirements Engineering. ContextUML and the ContextServ platform comprise the design and implementation phases of a software development process. One of the important open research issues remains: How can we inject context-awareness into the initial requirements engineering phase? One line of research on this issue is initiated by Ali et al. [2] where the authors propose a contextual goal modelling framework to derive goal model variants that meet the goals in a given context. Later on, in [3], the framework integrates the detection of both the context specification inconsistencies and goal conflicts resulting from variabilities. Recently, based on the above work, Botangen et al. [11] propose an approach for context-based requirements variability analysis in the goal-oriented requirements modelling where contextual goals and contextual preferences can be defined to specify the relationships of contexts with requirements and preferences. Future research questions include how to optimize the automated conflict detection algorithm and how to deal with evolving contextualization derived from the ever-changing nature of requirements.

Context-Aware Services Recommendation. In the era of information explosion, the number of Web services also increases rapidly, which brings new challenges for users to choose the right Web services among tens of thousands candidates. The research field of context-aware service recommendation aims to recommend services to users based on their contextual information. In [37], a time-aware service recommendation approach that integrates temporal information with content similarity is proposed. In [14], a two-level topic model that combines service content and service social network information is proposed for Web API recommendation. Recently, Botangen et al. [10] propose a geographic-aware collaborative filtering approach for Web services recommendation. Open research questions include how to exploit the recently flourishing deep learning methods in context-aware service recommendation, and how to design a generic

ensemble architecture to facilitate the integration of different types of contextual information.

Security and Privacy on Context-Aware Services. Security and privacy are the serious challenges for CASs, which need to be addressed for users to fully embrace the services. Contextual information often is related to sensitive personal data such as activities, transactions, and whereabouts. With embedding sensing being more and more prevalent on personal devices, personal sensing can be used to detect users' physical activities and bring privacy concerns. Building a trusted ecosystem among context-aware services requires appropriate measures on security and privacy between CASs, IoT devices that provide contextual information, and their interactions with service users [8]. Unfortunately, security and privacy are still not adequately addressed by the majority of existing approaches. The Blockchain technology has the potential to address these issues but need to consider several challenges such as resource limitation and low transmission rates of IoT devices. We believe that intensive research and development are needed in order to realize secure and trustworthy context-aware services.

Context-Aware Services in Mobile Edge Computing. With the proliferation of IoT and mobile devices, more and more services are moving towards the network edge, which minimizes the need on data transfers and reduces the latency. Mobile edge computing (MEC) has emerged as a key technology to assist wireless networks with cloud computing like capabilities, offering low-latency and mobility support services directly from the network edge. However, the dynamic and complex environment of MEC makes context-aware and adaptive predicting QoS of services a challenging task. In a recent work by Liu et al. [22], two context-aware QoS prediction schemes are proposed by considering user-related and service-related contextual information and MEC service scheduling scenarios. More research efforts are needed in this important direction.

5 Conclusions

Over the recent years, context-aware services (CASs) are emerging as an important technology for building innovative smart applications. Unfortunately, despite of active research and development, CASs are still difficult to build, due to lack of context provisioning management approach and lack of generic approach for formalizing the development process. In this paper, we have introduced the ContextServ project that focuses on developing a platform for model-driven development of CASs. We also review some representative research efforts on CASs in the literature and identify several open research issues that we wish to stimulate further research in this important area.

Acknowledgments

The ContextServ project has been partially supported by an Australian Research Council (ARC) Discovery Project grant DP0878367. Quan Z. Sheng's research has been also partially supported by an ARC Future Fellowship FT140101247.

References

1. Abowd et al, G.D.: Context-Aware Computing. IEEE Pervasive Computing 1(3), 22–23 (2002)
2. Ali, R., Dalpiaz, F., Giorgini, P.: A goal-based framework for contextual requirements modeling and analysis. Requirements Engineering 15(4), 439–458 (2010)
3. Ali, R., Dalpiaz, F., Giorgini, P.: Reasoning with contextual requirements: Detecting inconsistency and conflicts. Information and Software Technology 55(1), 35–57 (2013)
4. Aljubairy, A., Zhang, W.E., Sheng, Q.Z., Alhazmi, A.A.F.: SIoTPredict: A Framework for Predicting Relationships in the Social Internet of Things. In: Proc. of the 32nd International Conference on Advanced Information Systems Engineering (CAiSE 2020). pp. 101–116. Springer, Grenoble, France (2020)
5. Ayed, D., Taconet, C., Bernard, G., Berbers, Y.: CADeComp: Context-aware deployment of component-based applications. Journal of Network and Computer Applications 31(3), 224–257 (2008)
6. Badidi, E., Atif, Y., Sheng, Q.Z., Maheswaran, M.: On Personalized Cloud Service Provisioning for Mobile Users using Adaptive and Context-aware Service Composition. Computing 101(4), 291–318 (2019)
7. Benatallah, B., Dumas, M., Fauvet, M.C., Rabhi, F.A., Sheng, Q.Z.: Overview of Some Patterns for Architecting and Managing Composite Web Services. ACM SIGecom Exchanges 3(3), 916 (2002)
8. Bertino, E., Choo, K.R., Georgakopoulos, D., Nepal, S.: Internet of Things (IoT): Smart and Secure Service Delivery. ACM Transactions on Internet Technology 16(4), 22:1–22:7 (2016)
9. Botangen, K.A., Yu, J., Han, Y., Sheng, Q.Z., Han, J.: Quantifying the Adaptability of Workflow-based Service Compositions. Future Generation Computer Systems 102, 95–111 (2020)
10. Botangen, K.A., Yu, J., Sheng, Q.Z., Han, Y., Yongchareon, S.: Geographic-aware collaborative filtering for web service recommendation. Expert Systems with Applications 151, 113347 (2020)
11. Botangen, K.A., Yu, J., Yeap, W.K., Sheng, Q.Z.: Integrating context to preferences and goals for goal-oriented adaptability of software systems. The Computer Journal (2020)
12. Boudaa, B., Hammoudi, S., Mebarki, L.A., Bouguessa, A., Chikh, M.A.: An Aspect-oriented Model-driven Approach for Building Adaptable Context-aware Service-based Applications. Science of Computer Programming 136, 17–42 (2017)
13. Bouguettaya, A., Singh, M.P., Huhns, M.N., Sheng, Q.Z., Dong, H., Yu, Q., Neiat, A.G., Mistry, S., Benatallah, B., Medjahed, B., Ouzzani, M., Casati, F., Liu, X., Wang, H., Georgakopoulos, D., Chen, L., Nepal, S., Malik, Z., Erradi, A., Wang, Y., Blake, M.B., Dustdar, S., Leymann, F., Papazoglou, M.P.: A Service Computing Nanifesto: the Next 10 Years. Communications of the ACM 60(4), 64–72 (2017)
14. Cao, B., Liu, X., Rahman, M.M., Li, B., Liu, J., Tang, M.: Integrated content and network-based service clustering and web apis recommendation for mashup development. IEEE Transactions on Services Computing (2017)
15. Dey, A.K., Mankoff, J.: Designing Mediation for Context-aware Applications. ACM Transactions on Computer-Human Interaction 12(1), 53–80 (2005)
16. Hirschfeld, R., Costanza, P., Nierstasz, O.: Context-Oriented Programming. Journal of Object Technology 7(3), 125–151 (2008)

17. Hoyos, J.R., García-Molina, J., Botía, J.A.: A domain-specific language for context modeling in context-aware systems. Journal of Systems and Software 86(11), 2890–2905 (2013)
18. Hoyos, J.R., García-Molina, J., Botía, J.A., Preuveneers, D.: A model-driven approach for quality of context in pervasive systems. Computers & Electrical Engineering 55, 39–58 (2016)
19. Julien, C., Roman, G.C.: EgoSpaces: Facilitating Rapid Development of Context-Aware Mobile Applications. IEEE Transactions on Software Engineering 32(5), 281–298 (2006)
20. Kapitsaki, G.M., Kateros, D.A., Prezerakos, G.N., Venieris, I.S.: Model-driven development of composite context-aware web applications. Information and Software Technology 51(8), 1244–1260 (2009)
21. Kapitsaki et al., G.: Context-aware Service Engineering: A Survey. Journal of Systems and Software 82(8), 1285–1297 (2009)
22. Liu, Z., Sheng, Q.Z., Xu, X., Chu, D., Zhang, W.E.: Context-aware and Adaptive QoS Prediction for Mobile Edge Computing Services. IEEE Transactions on Services Computing pp. 1–1 (2019, Early Access)
23. Mo, J.P.T., Sheng, Q.Z., Li, X., Zeadally, S.: RFID Infrastructure Design: A Case Study of Two Australian RFID Projects. IEEE Internet Computing 13(1), 14–21 (2009)
24. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: State of the Art and Research Challenges. Computer 40(11), 38–45 (2007)
25. Prezerakos, G.N., Tselikas, N., Cortese, G.: Model-driven Composition of Context-aware Web Services Using ContextUML and Aspects. In: Proc. of the 5th International Conference on Web Services (ICWS'07). pp. 320–329 (2007)
26. Qin, Y., Sheng, Q.Z., Falkner, N.J.G., Dustdar, S., Wang, H., Vasilakos, A.V.: When Things Matter: A Survey on Data-centric Internet of Things. Journal of Network and Computer Applications 64, 137–153 (2016)
27. Sheng, M., Qin, Y., Yao, L., Benatallah, B. (eds.): Managing the Web of Things: Linking the Real World to the Web. Morgan Kaufmann (2017)
28. Sheng, Q.Z., Benatallah, B.: ContextUML: A UML-Based Modeling Language for Model-Driven Context-Aware Web Service Development. In: Proc. of the 4th International Conference on Mobile Business (ICMB'05). pp. 206–212. Sydney, Australia (2005)
29. Sheng, Q.Z., Pohlenz, S., Yu, J., Wong, H.S., Ngu, A.H., Maamar, Z.: ContextServ: A Platform for Rapid and Flexible Development of Context-Aware Web Services. In: Proc. of the 31st International Conference on Software Engineering (ICSE'09). pp. 619–622. Vancouver, Canada (2009)
30. Sheng, Q.Z., Qiao, X., Vasilakos, A.V., Szabo, C., Bourne, S., Xu, X.: Web Services Composition: A Decade's Overview. Information Sciences 280, 218–238 (2014)
31. Sheng, Q.Z., Yu, J., Dustdar, S. (eds.): Enabling Context-Aware Web Services: Methods, Architectures, and Technologies. CRC Press (2010)
32. Sindico, A., Grassi, V.: Model Driven Development of Context Aware Software Systems. In: International Workshop on Context-Oriented Programming (COP '09). pp. 7:1–7:5. New York, NY, USA (2009)
33. Tran, N.K., Sheng, Q.Z., Babar, M.A., Yao, L.: Searching the Web of Things: State of the Art, Challenges, and Solutions. ACM Computing Surveys 50(4), 55:1–55:34 (2017)
34. Yu, J., Han, J., Sheng, Q.Z., Gunarso, S.O.: PerCAS: An Approach to Enabling Dynamic and Personalized Adaptation for Context-Aware Services. In: Liu, C.,

Ludwig, H., Toumani, F., Yu, Q. (eds.) Proc. of the 10th International Conference on Service-Oriented Computing (ICSOC 2012). pp. 173–190. Springer, Shanghai, China (2012)
35. Yu, J., Sheng, Q.Z., Swee, J.K., Han, J., Liu, C., Noor, T.H.: Model-driven development of adaptive web service processes with aspects and rules. Journal of Computer and System Sciences 81(3), 533–552 (2015)
36. Yu, Q., Liu, X., Bouguettaya, A., Medjahed, B.: Deploying and Managing Web Services: Issues, Solutions, and Directions. The VLDB Journal 17(3), 537–572 (2008)
37. Zhong, Y., Fan, Y., Huang, K., Tan, W., Zhang, J.: Time-aware service recommendation for mashup creation in an evolving service ecosystem. In: 2014 IEEE International Conference on Web Services. pp. 25–32. IEEE (2014)