Computing (2018) 100:353–368
https://doi.org/10.1007/s00607-018-0602-0

A new paradigm of software service engineering in big data and big service era

Xiaofei Xu[1] · Gianmario Motta[2] · Zhiying Tu[1] · Hanchuan Xu[1] · Zhongjie Wang[1] · Xianzhi Wang[1]

Received: 22 February 2018 / Accepted: 28 February 2018 / Published online: 23 March 2018

Abstract In the big data era, servitization becomes one of the important development trends of the IT world. More and more software resources are developed and existed in the format as services on the Internet. These services from multi-domains and multi-networks are converged as a huge complicated service network or ecosystem, which can be called as Big Service. How to reuse the abundant open service resources to rapidly develop the new applications or comprehensive service solutions to meet massive individualized customer requirements is a key issue in the big data and big service ecosystem. Based on analyzing the ecosystem of big service, this paper presents a new paradigm of software service engineering, Requirement-Engineering Two-Phase of Service Engineering Paradigm (RE2SEP), which includes service oriented requirement engineering, domain oriented service engineering, and the development approach of software services. By means of the RE2SEP approach, the adaptive service solutions

✉ Xiaofei Xu
xiaofei@hit.edu.cn

Gianmario Motta
gianmario.motta@unipv.it

Zhiying Tu
tzy_hit@hit.edu.cn

Hanchuan Xu
xhc@hit.edu.cn

Zhongjie Wang
rainy@hit.edu.cn

Xianzhi Wang
xianzhi.wang@qq.com

1 Harbin Institute of Technology, Harbin, China

2 University of Pavia, Pavia, Italy

can be efficiently designed and implemented to match the requirement propositions of massive individualized customers in Big Service ecosystem. A case study of the RE2SEP applications, which is a project on citizens mobility service in smart city environment, is also given in this paper. The RE2SEP paradigm will change the way of traditional life-cycle oriented software engineering, and lead a new approach of software service engineering.

Keywords Software service engineering · Big service · Software reuse · Requirement pattern · Service pattern

Mathematics Subject Classification 68N01

1 Introduction

Recent advancements on Cloud Computing and Internet of Things (IoT) technologies propel the modern Internet towards a big data era [3]. The explosion of big data has strongly influenced on the rapid development of Internet of Services (IoS). The software services involved in processing big data have dramatically increased in both number and complexity. All these services from multi-domains and multi-networks are convergent, interrelated, and interoperated as a huge complicated service network. This service ecosystem can be called as *Big Service*.

We define *Big Service* as a massive, complicated series of services dealing with big data, which can be considered as a correlative complicated business in the networked virtual and real worlds [19]. Big Service has several critical new features: *Massiveness, Heterogeneity, Complexity, Convergence, Customer focus, Credibility and Value*, which are corresponding to the 5Vs (*Volume, Variety, Velocity, Veracity, Value*) features of big data. In big service ecosystem, there are extreme abundant *massive* software services on the Internet. These services show the *complex* forms with diversity, *heterogeneity* and multi-source of services, which are networked and interoperated across multi-domains, multi-networks, and multi-worlds. *Convergence* means the composite services are gathered, clustered and composited by multiple services to meet the customer requirements. *Credit* system is the basic precondition and foundation for constructing and operating the big service ecosystem. The ultimate goal of big service is to create *value*.

The ecosystem of big data and big service is changing not only the roles of software and services applications, but also the approaches to their development. The traditional software development approach, "from scratch" approach, is no longer in fashion. Maximal reuse of Internet accessible abundant open services sources has become the new highlight that can facilitate a rapid and efficient development of software and services. It provides agility in satisfying massive individualized customer requirements through on-demand application development. The researchers on software engineering and service engineering are exploring the new paradigm and more efficient approaches to develop or build the software services on-demand of massive individualized customers.

Through analyzing the features and phenomena of Big Service, we find that the customer requirements and the development of open, reusable service resources are key

factors to develop the new applications or comprehensive service solutions in the big data and Big Service ecosystem. The requirement engineering is used for presenting the proposition of customers service demand exactly. While the service oriented software engineering is applied to construct the service solutions based on open reusable software service sources, to meet the requirement proposition of customers. The domain oriented service patterns and features can be found through long term application experience of services and business in a certain domain. These service patterns can be used for constructing service solutions to meet the customer requirements. This paper focuses on a new paradigm of software service engineering for rapid development of service solutions in the Big Service ecosystem, in which the new approaches for service requirement engineering and domain-oriented software service engineering are emphasized. This new paradigm is called RE2SEP (Requirement-Engineering Two-Phase of Service Engineering Paradigm) in Big Service ecosystem.

In this paper, Sect. 2 shows the analysis of the related work; Sect. 3 presents a reference architecture of big service and analyzes the ecosystem of big service; Sect. 4 presents a framework of the RE2SEP paradigm; Sect. 5 presents service engineering and approaches in the RE2SEP paradigm; Sect. 6 shows a case study of RE2SEP applications; and Sect. 7 gives the conclusion.

2 Related work

Service Oriented Engineering (SOE) aims at agilely reusing legacy software systems to match various stakeholders requirements and interoperate their business processes. SOE has evolved and optimized the tradition process of software engineering, which follows a simple or repetitive top-down approach according to software lifecycle, such as the typical *waterfall model* [13], and even some other process models that introduces recursive or evolutionary process within the basic software lifecycle [15]. *Service Oriented Architecture* (SOA) is the most representative SOE methodology, which stemmed from *Component-Based Software Engineering* (CBSE) [10] and *Domain Engineering* [5]. As the service business and domain knowledge are quite hard to describe clearly and properly, so that Model Driven based and template based SOE methods became popular research topics for quite a long time. The following are the detail introduction of these methods.

CBSE is a software reuse-based approach that regards the definition of components as a starting point of engineering. By composing the components that cohesively implement different modules of a given system, the system can be finally obtained.

Domain Engineering aims at reusing concepts and implementations in previous software systems of the same domain, it represents the process of systematic software reuse where similar systems are repetitively built within a given domain. The systems share major commonalties yet have variations. The commonalities are often regarded as domain knowledge. Domain engineering requires analyzing and modeling the destination domain in advance. Domain engineering focuses on a family of systems rather than a single system.

SOA brings the servitization revolution of software engineering. IBM published *Service-Oriented Modeling and Architecture* (SOMA) as the first publicly announced

SOA methodology in 2004 [2]. Based on extending the OOP and CBSE methods, SOMA covers a broader scope and implements *service-oriented analysis and design* (SOAD) through the identification, specification and realization of services, components that realize those services (a.k.a. "service components"), and flows that can be used to compose services.

Model driven architecture (MDA) [12] is launched by Object Management Group (OMG), which is a type of forward engineering that produces code from human understandable specifications through model transformations from *Computation Independent Model* (CIM) to *Platform Independent Model* (PIM), and then to *Platform Specific Model* (PSM). Based on this concept, *Service Model Driven Architecture* (SMDA) [18] was proposed to provide a solution for establishing service systems responding to customer requirements in an agile style. In the SMDA approach, the MDA approach is applied for developing services, and the USML extended from UML is applied for describing service models. Multi-dimensional service model definition and transformation, and service components reuse are the main features of the SMDA.

In specific domains, priori knowledge can be modeled as templates, which can be used to business process rebuilding, system component reuse, code auto-generation, and etc. [16] introduces a method to use service template to describe service models, including product models, process models, resource models, and technique models. Becker et al. [1] proposes a configurative service engineering method, which defines a service template with configurable parameters, to automatically or manually match user requirement. In recent years, the service models and templates become more configurable, and context adaptable. For example, [17] proposes a BIRIS mode that plays as third-party board to aggregate heterogeneous services and individual user requirements. This mode can embed different service composition algorithms, such as QoS optimum based algorithm [8], context-aware based algorithm [14], ABC based algorithm [6], etc, to construct the most satisfied service solution.

As mentioned above, the SOE methodology has evolved more and more intelligent. However, in order to face the challenges of big service era [19], it must be able to use the standby services far more accurately, efficiently and rationally to build a complex, mature, and powerful software system to serve customers better. The service model/mode/pattern/template must have the self-learning ability, so that the SOE system/solution can meet the individual user requirements much more accurately. So, this paper proposes a new paradigm of Software Service Engineering. This paradigm will not only carry forward the achievements of DOE, CBSE and MDA based methods, but also promote the strengths of user requirement awareness and service system construction.

3 Big service ecosystem

3.1 A reference architecture of Big Service

A reference architecture of Big Service is shown in Fig. 1 [19], which consists of three core layers: *the local service layer, the domain-oriented service layer and the demand-oriented service solution layer*. Besides, this reference architecture includes

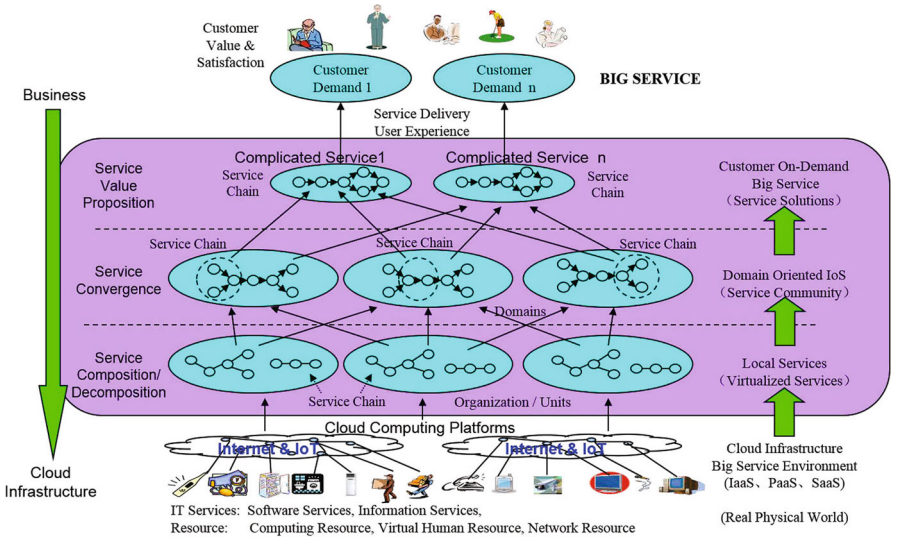

Fig. 1 The reference architecture of Big Service

two more layers: *the IT infrastructure layer* at the bottom and *the client layer* at the top.

The local service layer encapsulates the IT infrastructure, physical and digital resources into services, and organizes them together with the local services provided by individuals or organizations as fundamental services for Big Service. They include either atomic services or complex services composited by atomic ones. *The domain-oriented service layer* aggregates and composites the local services according to their related business domain, demands and relationships, forming a domain-oriented services communities or IoS. Services of all granularities are further linked through service chains or service hyper-chains across organizations, domains and networks to form a complex service ecosystem. This layer becomes the backbone of the Big Service ecosystem, which contains huge number of partial service solutions or service patterns in the service domains. *The demand oriented service solution layer* constructs customized service solutions by means of convergence of the domain-oriented services to meet the massive individualized customer requirements and to create value. *The IT infrastructure layer* at the bottom provides the basic condition and support for big services, while *the client layer* at the top delivers services to the customers based on customers requirements and value proposition.

3.2 Development of big service ecosystem

In the Big Service ecosystem, software services development environment is typically based on massive open service sources on the cloud. By virtualization philosophy computing infrastructure, platform, software, as well as applications are openly published and used as services. Moreover, there is an overwhelming trend of open APIs

in the industry, led by the growing availability of proprietary APIs. Besides, there are open APIs for IoT where various data collected by all types of sensors can be accessed through the Internet. To sum up, the available services and APIs have grown into a huge service repository. Some of them can be partially linked together based on business relevance, prior collaboration experience, and common standards, to become basic patterns for constructing more complicated service solutions.

In another end of the Big Service ecosystem, customers are enabled to present more and more individualized requirements for service systems. How to construct the adaptive service solutions efficiently and to response the massive individualized customer requirements rapidly become a big challenge. To match the complicated customer requirements with the adaptive service solutions is a critical factor of software service development and delivery.

The above changing trends lead to a new approach of software service development, in which developers pay more attention on proposition of customer requirements, construction of adaptive service solutions by means of reusing open service sources, and matching of requirements and service solutions. This new approach is defined as a new paradigm of software service engineering described as following sections.

4 RE2SEP: a new paradigm of software service engineering

Unlike the life-cycle oriented software engineering approaches, RE2SEP is a bidirectional approach with combination of two phases of service oriented requirement engineering (SORE) and domain-oriented service engineering (DOSE). The core technique of RE2SEP lies in not only the service oriented software engineering reusing the abundant open source in multi-domains/communities on the Internet, but also the matching between service requirements and service solutions using service context as a mediating facility.

The RE2SEP framework (shown in Fig. 2) consists of two different engineering processes with opposite directions. On the one hand, the DOSE process starts from available physical and virtualized software services in a certain domain and domain knowledge, and focuses on how to provide and leverage these services more sufficiently and wisely. In DOSE process, the services are organized in the format of service patterns to represent typical skeletons of service solutions or sub-solutions. Further, the service solutions can be constructed to meet the customer requirements, through service convergence from multi-domain service resource often based on service patterns. The context of service usage are often considered for service providers to deliver the adaptive service solutions to match the individualized customer requirements.

On the other hand, the SORE process begins with the individualized customer requirements and the typical virtualized partial user requirement description, and focuses on deriving application-specific service requirement propositions to build the adaptive service solutions. By analyzing massive individualized customer requirements, SORE is able to extract and define requirement patterns that frequently appear in customer requirements. These requirement patterns are generic and frequently occurring representation of fragments or pieces of user requirements in a certain application domain. They can be easily reused to form the customer requirement propositions,i.e.,

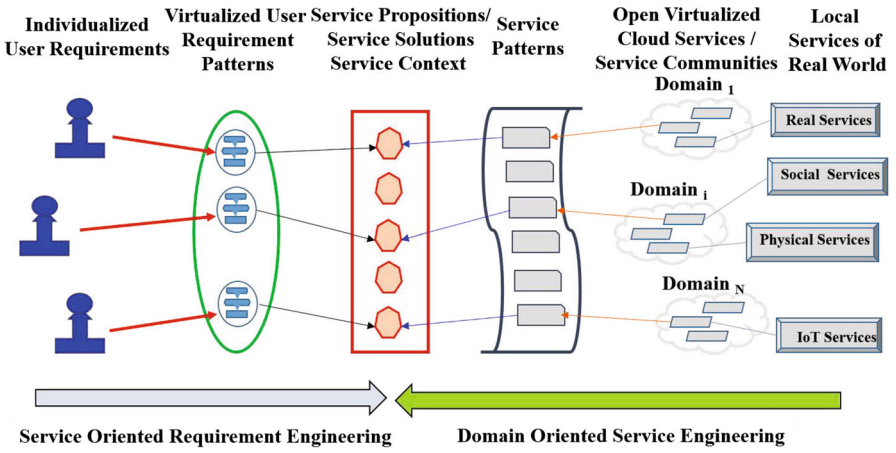

Fig. 2 The framework of the RE2SEP paradigm

every individual customer, the user requirement proposition would be related to a certain context of service usage which contain some requirement patterns.

The link of the user requirement propositions and the service solutions becomes the key interaction point of the two phases. To get the best matching of user requirement propositions and service solutions is the most important purpose of the RE2SEP paradigm. If requirement patterns and service patterns can be matched in some service contexts, then a mapping link between both sides is set up. Such mapping can be used to optimize the adaptive service solutions. Further, the decisions can be made on which service patterns to be selected or composited together to form the final service solutions, which fit for the most corresponding requirement patterns.

5 Approaches of software service engineering in the RE2SEP paradigm

5.1 Service oriented requirement engineering in the RE2SEP paradigm

The architecture of SORE is shown in Fig. 3. The main task of SORE is to acquire customers' massive and individualized requirements, and to describe the service requirement propositions formally. The objective is to facilitate representation and processing of the requirements, so as to enable effective service solutions recommendation.

The SORE is a top-down approach. The user requirements are initially presented by massive and individualized customers. These requirements can be classified and clustered according to the related business and the similarity in their specific specifications. Typical non-functional specification of service requirements include those related to Quality of Service (QoS), constraints, customer defined rules, customers preferences and value proposition regarding the expected service solutions. Customers value proposition illustrates how model can be extracted and well-organized as prior knowledge. These fragments, so called service requirement patterns, are introduced to

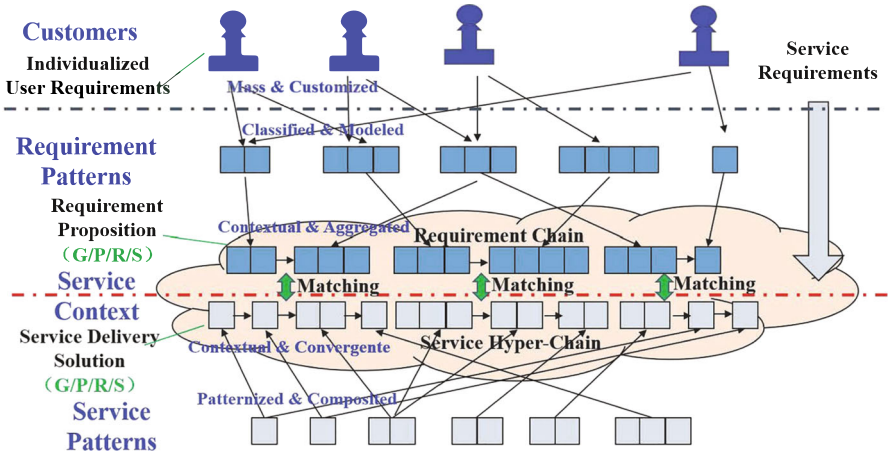

Fig. 3 Architecture of service oriented requirement engineering in the RE2SEP paradigm

represent the modularized pieces of description of customer requirements involving the usage experience of domain application. The service requirement patterns often share a similar or even identical mapping in the pool of service patterns defined in the service engineering process. The massive individualized requirements of customers can be aggregated into limited requirement patterns. Vice versa, through selecting and aggregating multiple related service requirement patterns, a specific requirement proposition can be presented precisely based on the service context of related application domains.

5.2 Domain oriented service engineering in the RE2SEP paradigm

The DOSE architecture is shown in Fig. 4. It is a bottom-up approach based on service patterns and open service sources, and is often facilitated by an integrated cloud service platform. According to the given service usage context, the service patterns are further linked together by means of service hyper-chain to form final service solutions.

In DOSE, service pattern can be either manually defined by domain experts or discovered automatically through data mining or statistics methods. A service pattern can be either typical independent business process or a partial service solution in the domain, like the role of composite components in building construction. However, the service patterns are more solution-oriented and business application-oriented.

As linkage among atomic services within a single service pattern, service chain is a higher-level concept of service sub-chain. In satisfying customers requirements at the application level, service patterns are selected and linked to form the final service solutions. The service patterns are further converged and linked together by service hyper-chain according to the application service context. In order to match the customer requirements precisely with service solutions, the mapping correlations among service patterns, service requirement patterns, requirement proposition and service solution as well as service context should be analyzed.

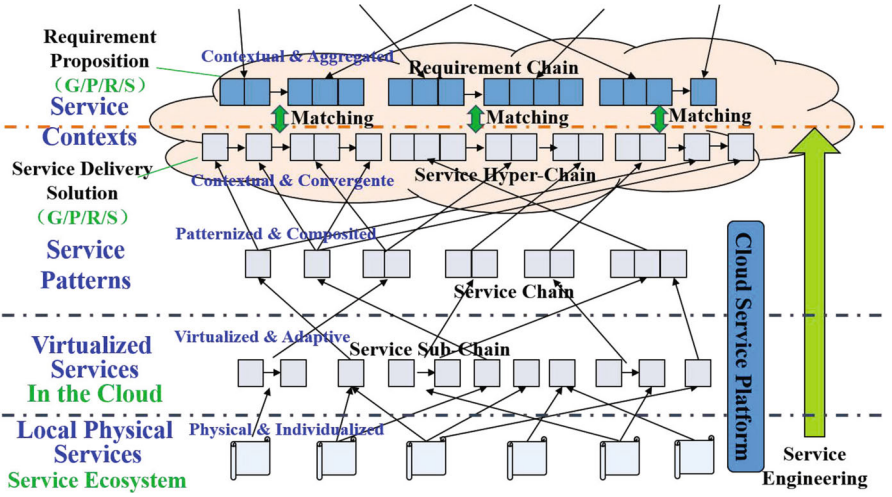

Fig. 4 Architecture of domain oriented service engineering in the RE2SEP paradigm

In RE2SEP paradigm, it is easy to find that the open software service resources, domain knowledge and priori knowledge about services usage history play very important roles in facilitating the software engineering process. The abundant reusable services and software resources are crucial and precondition to the RE2SEP paradigm, because they form the fundamental basis of the Big Service ecosystem. Only with sufficient available services, the service domains can be revealed in the service engineering process. The abundant services also make it possible to define and reuse the prior knowledge about a specific domain or a specific type of applications, thus improving the engineering efficiency and the quality of service solutions. In order to reuse the abundant open service sources in one and multi-domains, the virtualized services should be well structured and integrated into service patterns through service sub-chains, service chains and service hyper-chains in the Big Service ecosystem.

5.3 Development process of software services in RE2SEP paradigm

The development process of software services in the RE2SEP paradigm consists of five steps as follows:

(1) Construction of service development environment in the cloud platforms

Preparation of service development environment includes the facility for gathering and organizing real world services from one or multi-domains onto the cloud platform, virtualizing services, defining and structuring service patterns, etc. The prior knowledge of services usage in the domains is often used for construction of services or service patterns. The services are further improved or restricted by means of business rules and domain constraints. The services must be prepared and organized based on the experienced service solutions in the past and for the future expected possible service solutions.

(2) Presenting service requirements based on service requirement patterns

It can be found that even various massive customers have relative limited types of service requirements which are composition of requirement patterns formed through previous usage experience in the certain domains. The service requirement patterns are designed based on typical business usage manners in related certain domains, which can be easily matched by certain service patterns of the final service solutions. Some rules should be defined on which fragment of customer requirement can be replaced by requirement patterns, and how to deal with overlaps between different service requirement patterns.

(3) Service convergence based on open services resource and service patterns

Service convergence requires the compatibility among different services and service patterns from different sources, domains, organizations, and networks. Service convergence is targeted straightly at the typical solutions of service applications. It aims at providing some prepared composite service sets for typical solutions or sub-solutions directly based on the available services and service patterns in the service development environment before a specific actual requirement is known. Service chains and hyper-chains are designed to facilitate these typical solutions or sub-solutions consist of the converged service sets.

(4) Matching service requirements with service solutions

This step concerns matching between service requirements and service solutions at different granularities and aspects. A service solution could either be identified by selecting/modifying existing prepared services sets, or by combining existing/modified service patterns as newly constructed results. Service requirement patterns are matched with service patterns by using service context as mediation. Service context defines service usage information at different levels, such as how different requirement patterns and service patterns are mapped to each other, and how different service requirements can be satisfied by different available prepared service sets. For a certain service system, the result of matching can be evaluated in five aspects, i.e., matching efficiency, matching precision, matching effect, QoS of solutions, and choice scopes of matching.

(5) Service delivery and execution

The optimized service solutions would be delivered through the Internet of Services with the support of open service sources. In the Big Service ecosystem, the service delivery requires a cloud-based service running platform, a service center and service library, web portals or mobile user-end for interaction with customers. The service process planning, resource scheduling, and service execution are performed by a service execution engine on the cloud platform. The customers feedbacks and comments on the delivered service solutions can also be considered as evaluation results from the customer side. The performance and quality of services and service patterns may be monitored and evaluated during or after service execution. The feedback would be useful for improvement of Big Service ecosystem in the future.

6 IRMA project: a case study of RE2SEP

In order to understand the principles of the RE2SEP, a case study called IRMA (Integrated Real-time Mobility Assistant), is introduced, whose architecture and devel-

opment are close to the RE2SEP framework. IRMA is a project on mobility services in a smart city environment, developed by the Lab of Service Engineering of University of Pavia, Italy [11], and tested on the cities of Basiglio (a suburb of Milan) and Pavia with the cooperation of their respective municipalities.

6.1 Overview of IRMA

IRMA intends to manage the life-cycle of urban mobility on any combination of public / shared transports. Hence, IRMA uses not only static information such as maps and timetables, but also event information, such as real time position of transports, delays, alerts issued by transport authorities, and, last but not least, tweets from social network [9]. Such ecosystem of services and information serves the ecosystem of urban stakeholders, which includes (a) individual users (as citizens and tourist), who can use all available transports, (b) impeded users, who can use only some transports on some itineraries (elderly, wheel-chaired, and blind people), (c) transport providers and authorities (as public transport, taxi, shared transport providers, and, of course, municipalities) who manage and operate the transports. An overview of such ecosystem is in Table 1 that makes evident the wide range of stakeholders and of related services, which reflects the main objective of IRMA, i.e. mobility everywhere for everyone, thus including aspects, often neglected, like (a) disabled people, and (b) indoor mobility [7].

6.2 RE2SEP case description for trip planner of IRMA

One objective of IRMA project is to provide mobility to everyone everywhere, thus including also disabled people, and both outdoor and indoor mobility inside complex buildings. It defines a platform that fits small cities (as Pavia, with 70000 inhabitants and 2000 students), midsize cities (500,000 inhabitants) and large metropolitan areas with complex transport system (underground, buses, tramways, taxi, shared vehicles). We choose trip planner realized by the IRMA app for outdoor mobility as a case for RE2SEP. The layers and relations among service fragments are well illustrated in Fig. 5.

Actually, users, because of their different natural attributes (gender, age, nation, and etc.), have different behaviors and needs. Hence, some requirement fragments are common, while some are unique of a specific user group. By grouping users and refining related requirements, common/similar requirement fragments are extracted as requirement pattern. For example, the requirement fragment Get outdoor navigation is a common requirement, while Translation is a unique requirement that targets foreign visitors.

Meanwhile, there are abundant physical services (e.g. vehicles, traffic lights and WiFi sensors etc.) that have been prepared to fulfill the massive public requirements. Also, there are a large number of IoS services, i.e. web and mobile applications that provide information (e.g. maps, translation, and search) and social network (e.g. Google, Twitter etc.). IRMA converges all these physical services and IoS services into the cloud and makes them become accessible virtualized services. Some of these services

Table 1 An overview of IRMA service patterns

Category	Service	Service description
Services to individuals	Outdoor mobility	The flagship app, called trip planner, supports end to end trips in urban areas; it covers the whole itinerary cycle on public and shared transport systems; it defines, tracks real time and stores itineraries; it incorporates POIs (point of interest)
	Indoor mobility	It supports the whole life cycle of indoor mobility; differs from trip planner because maps are proprietary information and GPS is replaced by other technologies (wifi, guide objects, magnetic fields)
	Alert management	Alert and real-time mobility rescheduling in front of unexpected events. It links and filters social, web, crowd, collaborative and sensor data
Social services	Elderly	A version of trip planner for older people that can run also on smart TV, with usage of avatar animated menus, specific additional services as map of POIs, easy transports, safe ways, etc.
	Disabled	It addresses blind and wheel chaired people. It guides disabled people on a barrier-free itinerary. It includes indoor mobility, position of disabled people, healthcare services and transport provider. It implies the development of a smart stick with a precision around 1 cm
Services to municipalities	Analyzer	Analysis of flows of public transport, private vehicles and people (time series and real time); it links multiple data sources: social networks, crowd, collaborative applications, mobile network, transit sensors, noise sensors; it provides input to long and short term planning
	Short-term traffic forecast	It predicts the forthcoming traffic patterns (e.g. next half hour) in order to detect possible disruptions of mobility patterns

Table 1 continued

Category	Service	Service description
	Long-term traffic forecast	Long-term traffic forecast service to predict the impact on the city traffic implied by long-term modifications of the traffic flows in the area and also by planned events (shows, fairs, etc.)
	Emergency service planner	Planning/real-time rescheduling of mobility in front of unexpected events; uses data from multiple nodes of a transit network for enhancing reactive mobility services
	Air quality	It predicts the impact on air quality suffered by the neighbors due to construction activities and changes in traffic flows

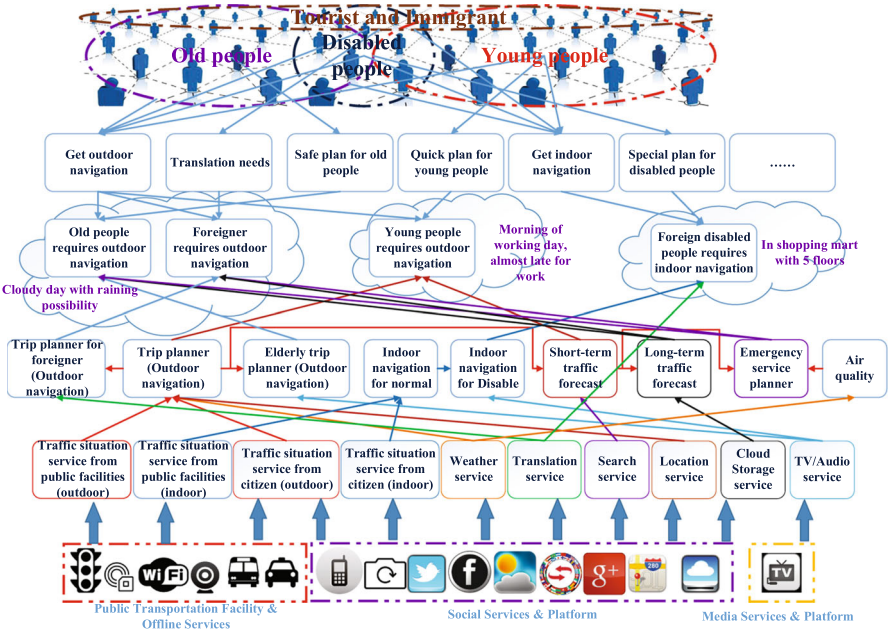

Fig. 5 Case description of RE2SEP in IRMA

have already been mashed up or composited as a new service, or can be combined to form a new service sub-chain. For example, traffic situation service from public facilities (outdoor) mashes several physical services from IoT devices, such as vehicles, traffic lights, WiFi sensors, cameras, bus, and etc. Thus, they can be treated as one service pattern as shown in Table 1.

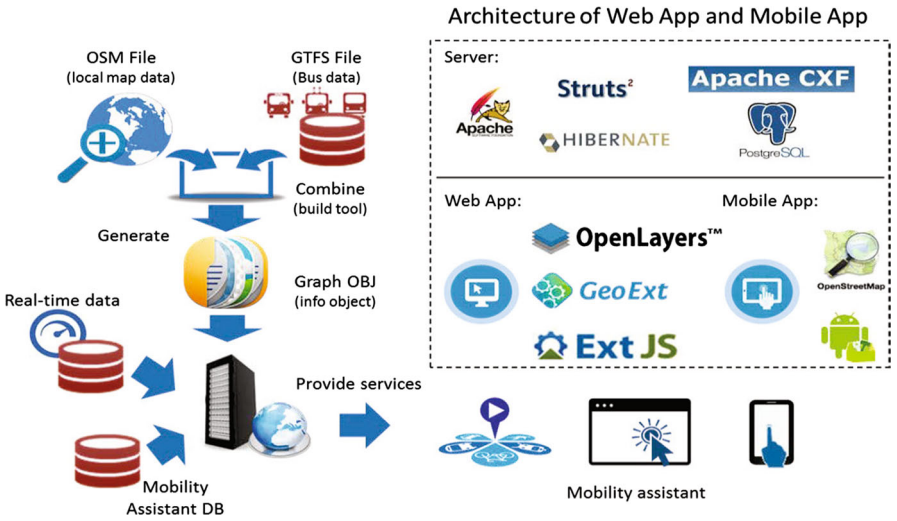

Fig. 6 The hierarchy of the trip planner service in the IRMA project

As mentioned in Sect. 4, individual customer's requirement proposition would be related to a certain context of service usage which contains some requirement patterns. For example, the requirement proposition, "old people requires outdoor navigation", consists of "get outdoor navigation pattern" and "safe plan for old people" pattern in common. If an old man has a travel plan on a cloudy day with raining possibility, "weather forecast" requirement would be necessary to "safe plan for old people" requirement pattern in this service context. In contrast, if a young people is almost late for work in a morning of working day, even though it is a rainy day, he will never care about the weather. Service context provides concrete information and constraints to support the matching between requirement patterns and service patterns. In this example, service patterns, "elderly trip planner (outdoor navigation)", "long-term traffic forecast", and "weather service", will be invoked simultaneously to serve the old man to make a safe travel schedule in the bad weather. And service patterns, "short-term traffic forecast" and "real-time outdoor navigation", will be invoked to serve the young man to find the fastest trip plan with less traffic jam, so that he can arrive his company in time.

6.3 Design and implementation

Trip Planner, which includes Web and Android applications, is built on a stack of Open Source platforms of the proposition (Fig. 6), namely:

- OSM (Open Street Map) that publishes map data;
- Time table, coded in GTFS, a template, proposed by Google, which describes relevant properties of transports;
- Real time data on the position of buses, that are published by the local Transport Authority, also coded according to a GTFS template for real time;

– The aggregate service, which combines static and real time data, and which is customer developed;
– The value proposition for citizens that runs on web and Android.

From Figs. 5 and 6, one can be easily elicited how the top down–bottom up approach works. The analysis cycle of the approach goes through stages that are alike the model driven architecture (MDA). At the outset, the designer must understand the needs of the individual who wants to move around on public transport. First he/she has to model the needs and related information requirements choose the transport, travel, capture alerts etc. by Use Case, Goal Oriented Analysis [4] or alike high level techniques. Then he/she has to define a model that is computer oriented, but still independent from implementation platforms, thus identifying the classes and sequence diagrams or other UML diagrams. At this point, he or she should search what kind of information sources and available services in lower level that can be used on the Internet. Afterwards, platform specific models must be defined, those in IRMA are all Open Sources.

7 Conclusion

More and more open and reusable service resources from multi-domains and multi-networks provide us a chance to apply these services into rapid development of the new applications or software services to meet massive individualized customer requirements. This paper presents a new paradigm of software service engineering, RE2SEP, in which the requirement patterns and service patterns are introduced for matching customer requirements with adaptive service solutions. By means of the open source based RE2SEP approach, the adaptive service solutions can be efficiently designed and implemented to meet the individualized customer requirements in Big Service ecosystem. IRMA, a case study of RE2SEP application, is given to show the advantage of this new approach.

Further work on RE2SEP will focus on detailed service engineering and methodology, including normalization of RE2SEP paradigm, service modelling, service pattern description, RE2SEP service solution engineering, service requirement analysis and modelling, service requirement pattern definition, service solution/service requirement matching, RE2SEP methodology and guideline, RE2SEP support tools and so on. More application case studies will be investigated and performed.

Acknowledgements Research work in this paper is supported by the Natural Science Foundation of China (Nos. 61272187, 61472106) and the Science and Technology Major Project of ShanDong Province (No. 2015ZDXX0201B02).

References

1. Becker J, Beverungen D, Knackstedt R, Matzner M (2009) Configurative service engineering—a rule-based configuration approach for versatile service processes in corrective maintenance. In: 42nd Hawaii international conference on system sciences, 2009 (HICSS'09). IEEE, pp 1–10
2. Bieberstein N, Laird R, Jones K, Mitra T (2008) Executing SOA: a practical guide for the service-oriented architect. Addison-Wesley, Reading

3. Dobre C, Xhafa F (2014) Intelligent services for big data science. Future Gener Comput Syst 37:267–281
4. Giorgini P, Rizzi S, Garzetti M (2005) Goal-oriented requirement analysis for data warehouse design. In: Proceedings of the 8th ACM international workshop on data warehousing and OLAP. ACM, pp 47–56
5. Harsu M (2002) A survey on domain engineering. Citeseer
6. Lartigau J, Xiaofei X, Nie L, Zhan D (2015) Cloud manufacturing service composition based on qos with geo-perspective transportation using an improved artificial bee colony optimisation algorithm. Int J Prod Res 53(14):4380–4404
7. Liu K, Motta G, Ma T (2016) Xyz indoor navigation through augmented reality: a research in progress. In: 2016 IEEE international conference on services computing (SCC). IEEE, pp 299–306
8. Liu ZZ, Jia ZP, Xue X, An JY (2015) Reliable web service composition based on qos dynamic prediction. Soft Comput 19(5):1409–1425
9. Ma T, Motta G, Liu K (2017) Delivering real-time information services on public transit, a framework. IEEE Trans Intell Transp Syst 18:2642–2656
10. McIlroy MD, Buxton J, Naur P, Randell B (1968) Mass-produced software components. In: Proceedings of the 1st international conference on software engineering, Garmisch Pattenkirchen, Germany, pp 88–98
11. Motta G, Sacco D, Ma T, You L, Liu K (2015) Personal mobility service system in urban areas: the IRMA project. In: 2015 IEEE symposium on service-oriented system engineering (SOSE). IEEE, pp 88–97
12. OMG (2001) OMG pursues new strategic direction to build on success of past efforts: model driven architecture. http://www.omg.org/news/releases/pr2001/2001-03-08a.htm
13. Royce WW (1970) Managing the development of large software systems. In: Proceedings of the IEEE WESCON, vol 26. Los Angeles, pp 328–338
14. Sheng QZ, Benatallah B (2005) Contextuml: a UML-based modeling language for model-driven development of context-aware web services. In: International conference on mobile business, 2005 (ICMB 2005). IEEE, pp 206–212
15. Sommerville I (2004) Software engineering. International computer science series. Addison Wesley, Reading
16. Wang J, Yu J, Han Y (2005) A service modeling approach with business-level reusability and extensibility. In: IEEE international workshop on service-oriented system engineering, 2005 (SOSE 2005). IEEE, pp 23–28
17. Wang Z, Xu X, Chu D, Mo T (2009) Architectural design of BIRIS-based marine logistics service platform and related interoperability issues. In: International conference on interoperability for enterprise software and applications China, 2009 (IESA'090), pp 149–156. IEEE
18. Xu X, Mo T, Wang Z (2007) SMDA: a service model driven architecture. In: Proceedings of the 3rd international conference on interoperability for enterprise software and applications, Funchal, Portugal. Springer, Berlin, pp 291–302
19. Xiaofei X, Sheng QZ, Zhang L-J, Fan Y, Dustdar S (2015) From big data to big service. IEEE Comput 48(7):80–83