Domain Priori Knowledge based Integrated Solution Design for Internet of Services

Hanchuan Xu, Xiao Wang, Yuxin Wang, Nan Li, Zhiying Tu, Zhongjie Wang, Xiaofei Xu

School of Computer Science and Technology, Harbin Institute of Technology

Harbin, China

Email: {xhc, wxlxq}@hit.edu.cn, 19B903081@stu.hit.edu.cn, {linan, tzy_hit, rainy, xiaofei}@hit.edu.cn

Abstract—Various types of services, such as web APIs, IoT services, O2O services, and many others, have flooded on the Internet. Interconnections among these services have resulted in a new phenomenon called "Internet of Services" (IoS). By IoS, people don't need to request multiple services by themselves to fulfill their daily requirements, but it is an IoS platform that is responsible for constructing integrated solutions for them. Since user requirements (URs) are usually coarse-grained and transboundary, IoS platforms have to integrate services from multiple domains to fulfill the requirements. Considering there are too many available services in IoS, a big challenge is how to look for a tradeoff between the construction efficiency and the precision of final solutions. For this challenge, we introduce a framework and a platform for transboundary user requirement oriented solution design in IoS. The main idea is to make use of domain priori knowledge derived from the commonness and similarities among massive historical URs and among historical integrated service solutions(ISSs). Priori knowledge is classified into three types: requirement patterns (RPs), service patterns (SPs), and probabilistic matching matrix (PMM) between RPs and SPs. A UR is modeled in the form of an intention tree (I-Tree) along with a set of constraints on intention nodes, and then optimal RPs are selected to cover the I-Tree as much as possible. By taking advantage of the PMM, a set of SPs are filtered out and composed together to form the final ISS. Finally, the design of a platform supporting the above process is introduced.

Keywords-Internet of Services; Integrated Service Solution; Domain Priori Knowledge; Bilateral Matching; Patterns

I. INTRODUCTION

In the big data era, servitization becomes one of the most important development trends in the IT world. More and more software services are developed and deployed on the Internet, along with a huge number of virtualized services that connect real-world physical service resources. Services from multiple domains, multiple networks, and multiple worlds are converged as a huge complicated service network or ecosystem, which can be called as "Internet of Services (IoS)" [1] or "Big Service" [2]. IoS presents a paradigm in which everything is available as a service on the Internet. In IoS, the extremely abundant massive services are diverse, distributed, and heterogeneous. By gathering, clustering, and composing these services, service solutions are produced to meet customer requirements. IoS is customer-focused so that when it receives a customers requirement, it creates integrated service solutions(ISSs) on demand, combining service resources to complete the service tasks. How to reuse the abundant service resources to rapidly develop new applications or ISSs to meet massive individualized customer requirements is the key issue in the IoS ecosystem.

Main technique challenges in IoS include:

- There are too many available services in IoS. A big challenge is how to look for a tradeoff between the construction efficiency and the precision of final solutions.
- Facing massive individualized customer requirements, how to assist users to present their requirements accurately in an understandable and convenient manner, and also for efficient construction of service solution is the key challenge in requirement elicitation and modeling.
- IoS is a far complex ecosystem that consists of various organizations and domains, multi-platforms, massive services, and users. How to design and implement a platform to support the efficient service solution construction and operation in IoS is a challenge.

Researchers on service engineering and service computing have proposed some service developing paradigms, such as Service Centric Systems Engineering (SeCSE) [3], Service Model Driven Architecture (SMDA) [4], Service-Oriented Modeling and Architecture (SOMA) [5] as well as thousands of approaches of service selection and composition [6] to address the issues of composite service design problem. There are also some researches for service mass customization [7], service network customization [8], and large-grained reuse of services [9]. However, there are still no sophisticated approaches to construct services accurately and efficiently, and developing service solutions to meet massive individualized customer requirements is still a very cumbersome and time-consuming activity, especially when these services are complex.

In our previous works [10], a new paradigm of software service engineering for the rapid development of ISSs in the Big Service ecosystem, whose name is RE2SEP (Requirement-Engineering Two-Phase of Service Engineering Paradigm) was proposed. One of the main ideas of RE2SEP is taking advantage of the domain priori knowledge of both the user side and the provider side to derive service patterns and requirement patterns. And then, the probabilistic matching matrices between the bilateral patterns are established to facilitate the service solution construction. Based on these previous works, we propose an ISS design framework and supporting platform of IoS in this paper, the overall architecture of which is shown
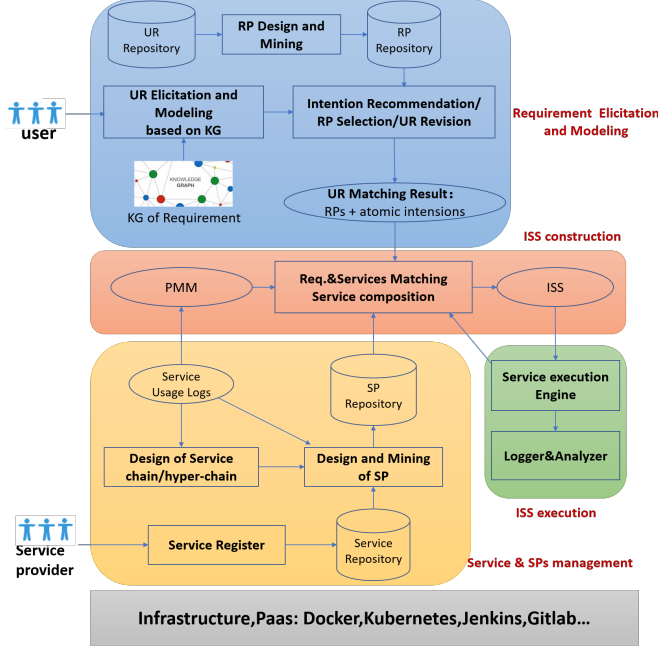
in Fig. 1.

Fig. 1. Framework and platform for ISS design of IoS

Main elements of the framework and platform, and contributions of this paper, are summarized as follows:

(1) Service Pattern Management: The priori domain usage knowledge of services is used for the construction of Service Patterns (SPs), which refer to typical complete or partial ISSs in a certain domain. The services repository and SP repository are well prepared for the future expected possible ISSs.

(2) Requirement Elicitation and Modeling: An intention tree model (I-Tree) is proposed, which supports end-users to present requirements in an understandable and convenient way. And then, a knowledge graph of requirements(KGR) is employed to assist users to complete requirement presentation automatically and quickly. It can be found that even various massive customers have relatively limited types of User Requirements (URs), which are the composition of Requirement Patterns (RPs) formed through previous usage experience in certain domains. RPs are used for recommendation and revision of the I-Tree. Finally, optimal RPs are selected to cover the I-Tree as much as possible.

(3) Integrated Service Solution Construction: A multidimensional probability matching matrix (PMM) is maintained which stores the matching information between SPs and RPs in different contexts. When a specific actual UR is coming, based on the optimal RPs selected in the step of requirement processing, by taking advantage of the PMM, a set of SPs are filtered out and composed together to form the final ISS efficiently and quickly.

(4) IoS Platform: The supporting platform provides supports to ISS design, development, and running in IoS. It adopts a decentralized and distributed architecture. It supports the organization of multi-layer services and SPs, and can support multi-layer and cross-domain service aggregation.

The remainder of this paper is organized as follows. Section II introduces requirement elicitation and modeling based on RP. Section III defines the SP. In Section IV, ISSs are constructed based on bilateral patterns and PMM. Section V presents the design of the IoS platform. Section VI introduces related work. Section VII concludes the paper.

II. PATTERN BASED REQUIREMENT ELICITATION AND MODELING

A. User Intention Tree

For end-users in a service system, expressing their requirements in natural language is the most natural and convenient way. However, this way usually leads to ambiguity, which is not conducive to the elicitation and analysis of requirements. In order to elicit and model requirements in an understandable manner and also in a professional well-defined pattern for developers, we propose an intention tree (I-Tree) model based on goal-oriented techniques that are commonly used in requirement engineering and have been advocated to express stakeholder objectives [11]. I-Tree model specifies the decomposition and constraint relationships among intentions. Fig. 2 shows elements and their relationships in the I-Tree meta-model.

As shown in Fig. 2, the I-Tree meta-model consists of the following elements: (1) User. (2) Role, which can be played by one or more users, and each user can play one or more roles.. (3) Intention, which describes specific functional requirements of users. (4) Decomposition, which represents a way to further decompose the upper intention into several lower intentions. There are two decomposition relationships between upper and lower intentions: AND and OR. The AND relationship means that the upper intention can be achieved only if all the lower intentions associated with it can be achieved. The OR relationship means that the upper intention can be achieved if any one of the lower intentions associated with it can be achieved. (5) Dependency. There are some dependencies between intentions, which means that the realization of one intention depends on another one. (6) Constraint, which denotes the description of the non-functional attributes of the intention. Constraint is defined in key-value pairs. i.e., $Constraint = < Cons_{name}, Cons_{type}, Cons_{value} >$. Constraints have different types, including enumeration, Boolean, and interval. It is necessary to distinguish different types of constraints because different types of constraints can be dealt with in different ways. (7) OptObjective, which refers to the user's preference for optimization objectives of the intention in ISS construction. The corresponding optimization strategy will be adopted to achieve the intention.

A practical example of I-Tree model for a wedding service requirement is shown in Fig. 3. The user plans to hold a wedding. The root intention includes the sub-intentions of the wedding banquet, wedding planning, guest pick-up, and inviting guests. The decomposition type between root intention and its sub-intentions is And. The overall optimization objective is the cost as low as possible. In the banquet intention, the user
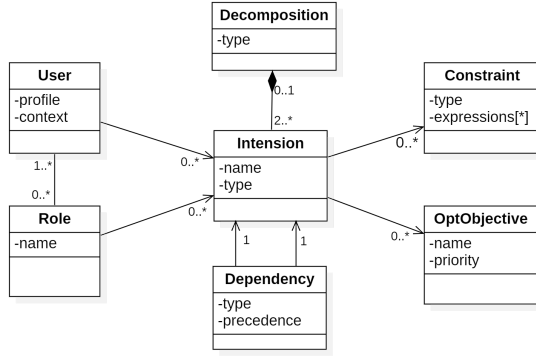
Fig. 2. Meta-model of intention tree (I-Tree)

specifies the constraint of the number of tables and decomposes fine-grained intentions: venue layout and food. Similarly, the figure shows users' intention decomposition, constraints, and optimization objectives for the wedding planning, inviting guests, and guest pick-up.

B. Requirement Pattern

It can be found that there is a set of requirements that always appear together in the user's intention trees. These requirements represent a common routine or relatively stable service process or sub-solution to be reused in satisfying different customer requirements in certain domains. We define these frequently occurring fragments of service requirement as RP, which is a modularized piece of the description of URs. RPs are expected to be reused in satisfying different URs. They are useful for presenting requirements in a quick and efficient way as well as for constructing ISS.

RP is defined as:

$$RP =< info, \{I - Tree\} >$$

where $info$ represents fundamental information of the RP, such as frequency of use, domain information, and description information. $\{I - Tree\}$ is the set of intentions along with their sub-intentions and constraints. An intention in an I-Tree is not only a node but also is a sub-tree. So an RP is a forest composed of I-Trees. In the example of the wedding UR shown in Fig. 3, there are two RPs, and one is composed of the intentions of inviting and pick-up guest, the other is the wedding banquet intention. RPs have various scales, and some complex ones are composed of I-trees, some may only have one single intention, so that stakeholders can flexibly model requirements via RPs.

To derive RPs from requirement repository, the first step is to mine the frequent substructures from I-Trees whose occur times reach a specific threshold. This can be abstracted as a frequent subgraph mining problem. We design an improved gSpan algorithm to deal with this issue. In the second step, the results, the frequent substructures, are grouped according to their functional requirements. Finally, in each group, intentions are clustered into RPs according to constraints. The constraints

of frequent substructures may be different within each group. The intentions are the same if the similarity between the constraints that exceeds the threshold. By comparison, frequent substructures can be divided into several classes according to the constraints. RP can be clustered from these similar frequent substructures by keeping common constraints and removing personalized ones. We will present the detailed RPs deriving method in another article.

C. Overall Process of Requirements

To address the challenge of the requirement process described in Section I, we first propose the I-Tree model to elicit and model URs clearly and conveniently. Then a KGR is generated and exploited to recommend related intentions to users for assisting them to complete the I-Tree automatically. Finally, RPs are selected to cover I-Trees. In the solution construction phase, corresponding SPs will be selected based on PPM to construct a solution meeting the URs quickly. The overall requirement process is shown in Fig. 4.

Step1: Requirement Elicitation and Modeling At the beginning of the requirement elicitation, users model I-Trees with their intentions and corresponding constraints, and initial I-Tree models are constructed. Its worth noting that there are two special situations of users' operation in the modeling process. One is that users may be reluctant to spend too much time filling in the intentions they need but don't care about. So they may not fill I-Trees in completely. The other is that users don't need some sub-intentions, so they also don't fill the sub-intentions into the I-Trees. It is hard to distinguish between these two situations. The rule we adopted is to consider only the intentions filled in by users, i.e., considering the second situation. In the step of intention recommendation described below, we will recommend intentions to users to help them complete the I-Trees automatically to deal with the first situation.

Step2: Intention Recommendation To assist users to model I-Trees, we design an intention recommendation algorithm based on the KGR. The intention recommendation algorithm can recommend frequently-used related intentions to users or help users fill in incomplete requirements by generating candidate intentions, just like the search candidates provided by a search engine.

The KGR is generated from URs stored in the requirement repository. When users enter requirements, the intention recommendation algorithm can deduce several possible intentions in the KGR based on users input and context information in I-Trees. Then the recommendation results are sorted according to specific rules, such as the similarity between requirements and user input, the use frequency of requirements. Finally, the sorted intention list become candidates provided to users for completing I-Trees.

Step3: Revision In the previous two steps, users fill in I-Trees based on their wishes and knowledge. They may not be familiar with the high-quality RPs and services that IoS platform can provide. So the platform may not be able to provide the best ISS based on the current I-Trees. For
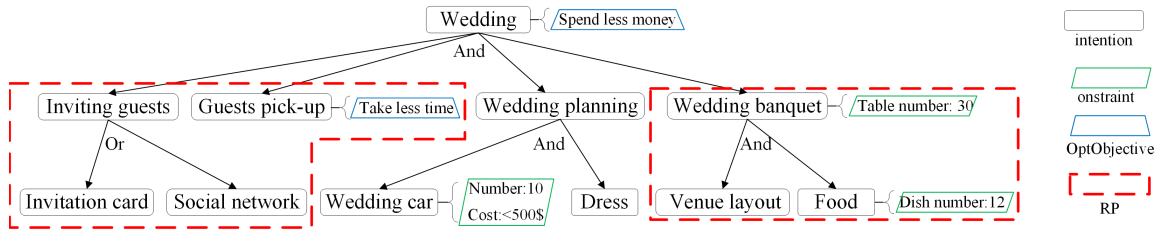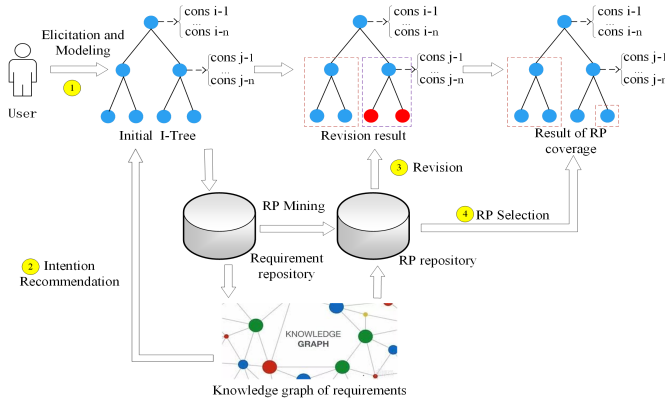
Fig. 3.   An example of I-Tree and RP

Fig. 4.   Requirement processing

such I-Trees, We design a revision approach to generate new similar ones by fine-tuning the users intentions according to the RPs with high popularity in the RP repository or relaxing the constraints. These new I-Trees will feedback to users as references. Users can modify the original I-Trees referring to them or accept the revision ones.

Step4: RP Selection After getting the final I-Tree, the RP selection algorithm is exploited to find the RPs, whose intentions can cover intentions of the I-Trees partially or completely. Here cover means that the intentions in RP are the same as the intentions of the I-Tree, and the constraints of the former are looser than the latter. The selection algorithm is run iteratively until any RPs cant cover the rest uncovered intentions of the I-Tree. There may be more than one selection schemes and the best one should be selected as the final selection result for generating an ISS. The rule is that the selection scheme with the highest coverage of the I-Tree and the smallest number of RPs is the best.

III. UNIFIED SERVICE SPECIFICATION AND SERVICE PATTERN

A. Service Pattern

Service pattern(SP) refers to a complete or partial ISS that is frequently used in a specific domain and has a business process, service activities, and service resources.

There are three essential elements of SP. The first is a description of the service process. The second is a set of service instances, which is related to the service business process and can be used to instantiate the SP. The last is the

verifying degree of the SP, that is, the evaluation index of SP. It includes the reusability of SPs and the promotion effect of optimization efficiency like increasing the matching speed or enhancing the optimization objectives., etc.

SP reflects the apriority of service in business and optimization. It is a service complex composed of behavior based on business association, and it is a combination in essence. SPs are often extracted by analyzing a large number of business experiences, and they can also be defined by domain experts. We can formally define an SP as:

$$SP =< info, fr, process, QoS, cons, instances >$$

where $info$ is the basic information of the SP, fr is the functional requirement of the SP, $process$ is used to describe the service process information of the SP, we use BPMN (Business Process Model and Notation) 2.0 XML format to describe and store the service process, QoS is the quality information of the SP, $cons$ is the execution constraint information of the SP, and $instances$ is the instance set of the SP.

For example, taking Uber and taking a taxi are two services that have the same function: urban traffic. So they can be looked on as the same kind of service. Alike, traveling by train and traveling by air are also the same kind of service. Historical service usage data show that taking Uber and traveling by train are often composited together at the same time in an ISS to meet URs, so taking a taxi and traveling by air do. So <urban traffic, inter-city traffic> can be abstracted as an SP, and <taking Uber, traveling by train> and <taking a taxi, traveling by air> are two SP instances.

Compared with using atomic services, using SPs to construct ISSs has multiple advantages, such as to achieve fast matching of URs, to build ISSs quickly, and to improve the efficiency of service composition optimization to get better ISS. Due to the large granularity of SPs, an SP can meet the requirements of users in multiple service activities and can get ISSs faster and get better user satisfaction.

We use k-means clustering algorithm, and frequent subgraph mining approaches gSpan to derive SPs. Firstly, we group similar services into multi-dimensional groups according to the multi-dimensional similarity measurement of function, input and output, user group, and provider, and calculate the conditional association probability of each dimension group. Then, aiming at the characteristics of services, we use frequent sub-graph mining algorithm to recognize the high-frequency service segments. Finally, the SP is abstracted from

the SP instances by semi-supervised clustering to ensure that the instances in each class cluster have the same process workflow, and the service class in the instances are the same.

The recognition of SP focuses on two key indicators: granularity (the number of service activities included in SP) and usage frequency (support). Granularity determines the reusability of the SP. Small-grained SPs have higher reusability, but lower reuse value; large-granularity SPs have lower reusability, but high reuse value. Therefore, the granularity of SP needs to be balanced between reusability and reuse value. The usage frequency determines the possibility of using SPs, and the choice of frequency should also be balanced between the number of SPs and the frequency. Finally, we extract different granularity and frequency SPs to complete the fast matching of supply with demand.

IV. ISS CONSTRUCTION BASED ON BILATERAL PATTERNS

A. Problem Description and Model

The essence of ISS construction in IoS is a supply-demand problem, i.e., matching the requirements from users(demand side) and the supplies from service providers(supply side) fast and accurately, which is a typical complex optimization problem. Since the minimization problem and the maximization problem can be transformed equally, its general model can be described as follows:

$$\min F(\bar{x}) = (f_1(\bar{x}), \dots, f_p(\bar{x}))^T \tag{1}$$

$$\text{s.t.}$$

$$g_i(\bar{x}) \leq 0, \quad i = 1, \dots, k \tag{2}$$

$$h_j(\bar{x}) = 0, \quad j = k+1, \dots, m \tag{3}$$

$$\bar{x} \in R^n \tag{4}$$

where

- $F(\bar{x}) = (f_1(\bar{x}), \dots, f_p(\bar{x}))^T$ is objective function vector, i.e., p objectives to be optimized. When $p = 1$, It's a single-objective optimization problem. The optimization objectives of service systems usually include the shortest service completion time, the lowest user expenditure cost, the largest profit of the service provider, the largest resource utilization, the highest user satisfaction, and so on.
- $g_i(\bar{x}) \leq 0, (i = 1, \dots, k)$ and $h_j(\bar{x}) = 0, (j = k+1, \dots, m)$ are constraints. In general, the constraints come from the demand side and the service supply side. Demand deadlines, precedence constraints between service tasks, response time, demand budget, etc. are demand constraints. Service resource available quantity, supply modes(Reserved, On-demand, Spot), etc. are supply constraints.
- $\bar{x} \in R^n$ are decision variables.

B. ISS Construction Algorithms

ISS construction is a kind of very complex optimization problem with characteristics of large scale, multi objectives, and complex constraints. These problems are basically NP-hard problems. Under the existing conditions, it is difficult to design new algorithms and increase computing resources to solve the problem effectively. Our idea is modularizing fine-grained requirements and services into coarse-grained RPs and SPs based on priori domain knowledge and then using intelligent algorithms to reduce the scale of the original problem. The RPs and SPs reflect the fixed collocation of requirements and services. Therefore, a complex requirement can be decomposed into different sub-modules(RPs), and the whole problem is decomposed into some sub-problems. The sub-problems can be further decomposed until all requirements can be achieved by SPs or atomic services.

The SPs are composed of the services that frequently serve the requirements together, but the same RPs in different contexts and with different optimization objectives can be matched by different SPs. How to choose the appropriate SPs corresponding to the RPs is the core of the ISS construction, which needs to reflect the effectiveness and efficiency of the algorithm. Via decomposition into RPs, a large-scale problem with complex UR can be decomposed into some small-scale sub-problems to improve efficiency; meanwhile, SPs are the optimization results obtained through machine learning approaches from priori domain knowledge, and reasonable matching can ensure the effectiveness of the algorithm.

To further improve matching efficiency and success rate, we construct a probability matching matrix(PMM) to link RPs and SPs. The PMM is a description of the relationship between RPs and SPs, and constructed by summarizing successful experiences and domain knowledge. Because in different service contexts and for different optimization objectives, an RP can be matched by different SPs with different probabilities, the PMM is multi-dimensional, as shown in Fig. 5. The X-axis is the RPs. The Y-axis is the SPs, and the Z-axis is the context that describes the matching scenarios. Context includes the information of users, environment, and objectives.
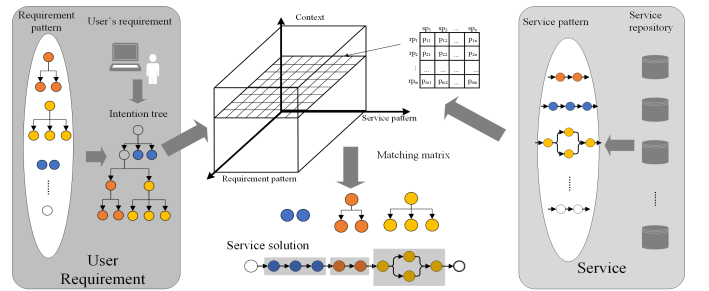

Fig. 5. The matching matrix (PMM) and matching process

The matching probability is calculated by a comprehensive evaluation of using times, matching quality results, matching difficulty, and other factors. PMM will be updated periodically.

We have designed and implemented a series of algorithms based on PMM for different contexts and optimization strategies. The algorithms are categorized for:

- Number of objective: *single*, *multi*

- Priority of solution quality and efficiency: *the optimal solution*, *the satisfactory solution*
- Service supply modes: *reserved*, *on-demand*, *spot*
- Algorithm strategy: *exact*, *rule-based*, *heuristic*, *meta-heuristic*

V. DESIGN AND IMPLEMENTATION OF IoS PLATFORM

IoS platform needs to aggregate various services from various organizations and domains. To this end, a decentralized and distributed architecture is adopted in the Big Service platform, which means each node has the same functions and implementations as peers.

IoS nodes use the P2P protocol for discovery and routing, and use inter-node communication to achieve cross-node requirement service matching and service solution execution. Platform users can post requirements, perform service matching, generate solutions, and execute on any node. Each node not only allows services to be deployed inside, but also allows registration of external services and third-party platforms.

The architecture of each IoS node is shown in Fig. 6, which can be divided into four layers.

A. Infrastructure Layer

Infrastructure Layer provides underlying supports such as container construction, management, and deployment in the cluster. We use Docker as the underlying containers of the platform and use Kubernetes to manage the containers. The Docker Registry, automatic image build tools(S2I and Jenkins), performance monitoring tools, and the services deployed on the IoS platform all run in the Kubernetes pods. Docker Registry is deployed in the cluster to store service images. The S2I and Jenkins provide two different ways to build Docker images from source code.

B. Service Management Layer

Service Management Layer provides supports for service management such as service registration, service quality management, service information caching, service log collection and processing.

C. Service Pattern Management & ISS Construction Layer

This layer consists of the multi-layer service pattern mining and storage tools, and ISS construction and execution engines.

As introduced above, SPs are harvested from priori domain knowledge. SPs are organized into three layers, i.e., the organization layer, the Inner-Domain layer, and the Cross-Domain layer. In the organization layer, we aggregate the cooperation relationships between the services provided by the same service provider. There are two types of entities, i.e., atomic services and service patterns. The relationships between entities in this layer are mainly cooperative relationships, which contain four sub-types as follows: symbiosis, parasite, location complementarity, and temporal complementarity. In the Inner-Domain layer, services and SPs from different organizations in the same domain are aggregated to a service chain that spans multiple organizations. Furthermore, after further grouping
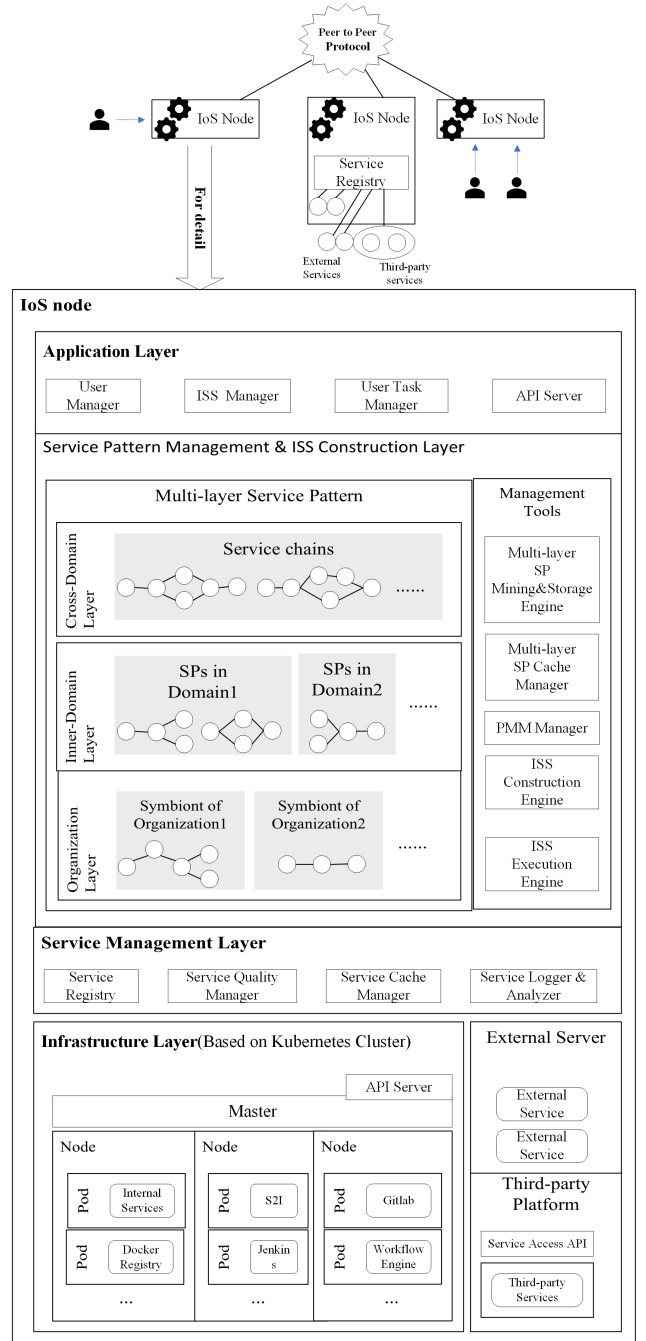

Fig. 6. Architecture of the platform

according to the service features conversion layer entities, the competitive relationship between the entities can be extracted. In the Cross-Domain layer, SPs that cross different domains are linked together to form a service super-chain, which corresponds to specific coarse-grained requirements.

D. Application Layer

We provide interactive interfaces and management applications to users. One example of ISS construction is shown in Fig. 7.
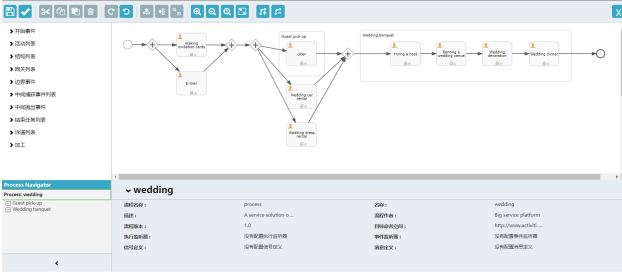
Fig. 7. System interface example: ISS construction

VI. RELATED WORK

A. Pattern-based Service Construction

Service patterns are gradually extended from traditional workflow patterns to service matching, and in most related studies, patterns are regarded as structured workflow or business process. [12] defines the SP and proposes that RPs are matched with SPs with service context as mediation. It is found that historical ISSs contains much priori service usage knowledge, which have a great impact on the construction of ISS, so mining patterns from historical ISSs with the data-mining techniques becomes the main approach [13]. [14] extracts the frequent process fragments in service composition through data-mining techniques, and use the fragments in service composition that followed. [15] uses the FP-growth mining method to mine valuable SPs from historical ISSs. These schemes try to automatically acquire the business patterns that are reused in history, identify the similar parts in structure and function, and combine them as high-level abstract patterns.

B. Goal-oriented Requirement acquisition

Requirement acquisition is a classic topic. Many scholars use ontology-based analysis to explore traditional requirements acquisition methods. [16] used ontology to obtain requirements for security software. [17] proposed an ontology-based approach to analyze the interaction between software and its interactive environment to capture software requirements. With the accumulation of data, many scholars use the method of data analysis to construct user portraits to predict user requirements. [18] adopts the method of weakly supervised learning to extract attribute labels from Twitter social data to construct users' portraits. Besides, much research has been done by extracting entities and generating associations between entities and users' interests to realize the focus mining of domain users [19].

C. Related Frameworks and Comparison

Because the core idea of this paper is to mine bilateral patterns, RPs and SPs, from priori domain knowledge for reuse in a large-grained way, several related frameworks based on reuse techniques are selected for comparison. Software Product Line (SPL) engineering refers to the engineering and management techniques to create, evolve, and sustain a

software product line which is a portfolio of similar software-based systems and products produced from a shared set of software assets using a common means of production [20]. Reuse-based Software Engineering (RBSE) promotes productivity by avoiding redevelopment and improves quality by integrating components that have established reliability [21] [22]. Service-Oriented Modeling and Architecture(SOMA) implements a component-based software engineering approach to service-oriented analysis and design, including service identification, service definition, and service implementation [5]. Service Model Driven Architecture(SMDA) helps service engineers describe the design and implementation of service systems that meet user service requirements through a model-driven approach [4]. Service-Oriented Development In a Unified framework (SODIUM) presents a model for developing service applications, focusing on how to define new services using coarse-grained combinations of services that can be reused [23].

We compare the related frameworks with our proposed framework in the following aspects:1) Service Constructon Stages, how the services are constructed. 2) Requirement Utilization, in requirement elicitation and modeling, what kinds of requirements are employed, i.e., atomic or/and composite requirements(RP). 3) Service Utilization, in service selection and composition, what kinds of services are employed, i.e., atomic or/and composite services(SP). 4) Domain Service Supported, whether supporting the construction for single-domain servcies or/and for cross-domain services. The comparison result is shown in Table I.

The service construction process of RBSE, SPl and our proposed framework can be divided into three main stages. The first stage is called Requirement Engineering(RE), which is a top-down process and elicitating and modeling URs. The Second stage is the Domain Engineering(DE), which is a bottom-up process with the task of finding and developing reusable services. The last stage is to construct service solutions based on results of RE and DE. Other framworks are top-down one-way processes. With the With the bidirectional strategy of RE-DE, domain knowledge and service assets can be better utilized on both the demand side and the service provsion side. Moreover, only our framework provides the supports of mining and utilizaiton of bilateral patterns, which can make full use of priori domain knowledge on both the demand side and the service side to improve the construction efficiency and user satisfaction of service solutions. It is worth pointing out that RPs and SPs can be used in other frameworks, we assign the values of those fields to star to indecate this in Table I. Last, as described in Section V-C, SPs are harvested from priori domain knowledge within organizations, within domains, and across domains, so our proposed framework supports the construction of cross-domain services. To conclude, the proposed framework makes better use of domain priori knowledge derived from the commonness and similarities among massive historical URs and among historical ISSs, which leads to it can provide supports for the quick and accurate construction of ISSs.

TABLE I
COMPARISON OF DIFFERENT SERVICE DEVELOPING PARADIGMS

Frameworks	Service Construction Stages	Requirement Utilization		Service Utilization		Domain Service Supported	
		atomic Req.	RP	atomic service	SP	single-domain	cross-domain
RBSE	RE-DE,top-down&bottom-up	+	*	+	*	+	-
SPL	RE-DE,top-down&bottom-up	+	*	+	*	+	-
SOMA	top-down	+	*	+	*	+	-
SMDA	multi-stages,top-down	+	-	+	-	+	-
SODIUM	multi-stages,top-down	+	-	+	-	+	-
The proposed framework	RE-DE,top-down&bottom-up	+	+	+	+	+	+

VII. CONCLUSION

To quickly and efficiently construct ISSs in IoS, we present a service designing framework and supporting platform of IoS. The framework takes advantage of priori usage knowledge of both the customer side and the provider side to derive SPs and RPs. And then, the probabilistic matching matrices between bilateral patterns are established. When a new requirement is coming, designers or customers can present it easily and explicitly with the proposed intention tree model and RPs. Subsequently, based on the bilateral patterns and their matching matrices, ISSs can be gotten more efficiently than traditional approaches. Finally, the introduced platform can provide the necessary management and running supports for the framework.

Our ongoing and future work includes: (1) to design more supply-demand matching optimization algorithms to support the service construction with various optimization objectives. (2) More application case studies will be investigated and performed to verify the effectiveness of the framework and the performance of the platform.

REFERENCES

[1] "Towards the internet of services," *Community Research and Development Information Service, European Commission*, Apr. 2014;https://cordis.europa.eu/fp7/ict/ssai.

[2] X. Xu, Q. Z. Sheng, L.-J. Zhang, Y. Fan, and S. Dustdar, "From big data to big service," *Computer*, no. 7, pp. 80–83, 2015.

[3] M. Di Penta, L. Bastida, A. Sillitti, L. Baresi, N. Maiden, M. Melideo, M. Tilly, G. Spanoudakis, J. G. Cruz, J. Hutchinson *et al.*, "Secseservice-centric system engineering: An overview," *At your service: service-oriented computing from an EU perspective*, pp. 241–272, 2009.

[4] X. Xu, T. Mo, and Z. Wang, "Smda: A service model driven architecture," in *Enterprise Interoperability II*. Springer, 2007, pp. 291–302.

[5] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley, "Soma: A method for developing service-oriented solutions," *IBM systems Journal*, vol. 47, no. 3, pp. 377–396, 2008.

[6] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decades overview," *Information Sciences*, vol. 280, pp. 218–238, 2014.

[7] S. A. Brax, A. Bask, J. Hsuan, and C. Voss, "Service modularity and architecture–an overview and research agenda," *International Journal of Operations & Production Management*, 2017.

[8] Z. Wang, X. Xu, and X. Wang, "Mass customization oriented and cost-effective service network," in *International IFIP Working Conference on Enterprise Interoperability*. Springer, 2013, pp. 172–185.

[9] Y. Rong, L. Bing, W. Jian, Z. Li, and H. Yan, "Reusing service process fragments with a consensus between service providers and users," *Chinese Journal of Electronics*, vol. 25, no. 4, pp. 648–657, 2016.

[10] X. Xu, G. Motta, Z. Tu, H. Xu, Z. Wang, and X. Wang, "A new paradigm of software service engineering in big data and big service era," *Computing*, vol. 100, no. 4, pp. 353–368, 2018.

[11] J. Horkoff and E. Yu, "Interactive goal model analysis for early requirements engineering," *Requirements Engineering*, vol. 21, no. 1, pp. 29–61, 2016.

[12] X. Xu, R. Liu, Z. Wang, Z. Tu, and H. Xu, "Re2sep: A two-phases pattern-based paradigm for software service engineering," in *2017 IEEE World Congress on Services (SERVICES)*, 2017.

[13] X. Wang, W. Niu, L. Gang, X. Yang, and Z. Shi, "Mining frequent agent action patterns for effective multi-agent-based web service composition," in *Agents and Data Mining Interaction - 7th International Workshop on Agents and Data Mining Interation, ADMI 2011, Taipei, Taiwan, May 2-6, 2011, Revised Selected Papers*, 2011.

[14] B. Upadhyaya, R. Tang, and Y. Zou, "An approach for mining service composition patterns from execution logs," *Journal of Software: Evolution and Process*, vol. 25, no. 8, pp. 841–870, 2013.

[15] R. Liu, X. Xu, Z. Wang, Q. Z. Sheng, and H. Xu, "Probability matrix of request-solution mapping for efficient service selection," in *2017 IEEE International Conference on Web Services (ICWS)*, 2017.

[16] A. Souag, C. Salinesi, R. Mazo, and I. Comyn-Wattiau, "A security ontology for security requirements elicitation," in *International symposium on engineering secure software and systems*. Springer, 2015, pp. 157–177.

[17] X. Chen and Z. Jin, "Capturing requirements from expected interactions between software and its interactive environment: an ontology based approach," *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 01, pp. 15–39, 2016.

[18] J. Li, A. Ritter, and E. Hovy, "Weakly supervised user profile extraction from twitter," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 165–174.

[19] W. Qiang, J. Zhi, and X. Yan, "Service discovery for internet of things based on probabilistic topic model," *Journal of Software*, vol. 25, no. 8, pp. 1640–1658, 2014.

[20] M. C. Bastarrica, J. Simmonds, M. Marques, and P. O. Rossel Cid, "Software product line evolution: A systematic literature review," 2019.

[21] R. W. Selby, "Enabling reuse-based software development of large-scale systems," *IEEE Transactions on Software Engineering*, vol. 31, no. 6, pp. 495–510, 2005.

[22] A. Perucci, M. Autili, and M. Tivoli, "A multipurpose framework for model-based reuse-oriented software integration synthesis." in *MODELS (Satellite Events)*, 2017, pp. 38–44.

[23] S. Topouzidou, "Service oriented development in a unified framework (sodium)," Deliverable CD-JRA-1.1. 2, SODIUM Consortium (May 2007), Tech. Rep., 2007.