A graph-based context-aware requirement elicitation approach in smart product-service systems

The paradigm of Smart product-service systems (Smart PSS) has emerged recently owing to the edge-cutting Information and Communication Technology (ICT) and artificial intelligence (AI) techniques. The unique features of Smart PSS including smartness and connectedness, value co-creation and data-driven design manner, enable the collection and analysis of large volume and heterogeneous contextual data to extract useful knowledge. Therefore, requirement elicitation, as a critical process for new solution (i.e. product-service) design, can be conducted in a rather context-aware manner, assured by those massive user-generated data and product-sensed data during the usage stage. Nevertheless, despite a few works on semantic modelling, scarcely any reports on such mechanism in today's smart, connected environment. Aiming to fill this gap, for the first time, a graph-based context-aware requirement elicitation approach considering contextual information within the Smart PSS is proposed. It leverages the pre-defined product, service, and condition ontologies together with Deepwalk technique, to formulate those concepts as nodes and their relationships as the edge of the proposed requirement graph. Implicit stakeholder requirements within a specific context can be further derived based on such interrelationships in a data-driven manner. To demonstrate its feasibility and effectiveness, an example of smart bike share system is addressed to illustrate the requirement elicitation process. It is hoped that this explorative study can offer valuable insights for the service providers who would like to extract requirements not only from the voice of customers but also from the user-generated data and product-sensed data.

Keywords: product-service systems, requirement elicitation, information modelling, graph embedding, context-awareness, ontology

1 Introduction

Product-service system (PSS) has stepped into a new paradigm named Smart PSS, owing to the

rapid development of Information and Communication Technologies (ICT) (e.g. ubiquitous connectivity and the online-offline smartness (Zheng et al. 2018)). It enables ever smarter product-service solution bundles such as predictive maintenance (Szwejczewski, Goffin, and Anagnostopoulos 2015, Zheng et al. 2019a), remote monitoring (Grubic and Jennions 2018) and self-adaptability (Zheng, Chen and Shang 2019), to name a few, and accordingly facilitates more dynamic interactions among stakeholders. As a result, the number of product-service solutions are continuously growing, and they can be recommended to a high variety of stakeholders based on their own requirements. However, this can lead to a paradox that stakeholders may not know which solution suit them best based on their own usage context. Furthermore, stakeholder requirements vary frequently due to the change of such usage contexts/usage scenarios, which directly affect the performance of the product-service bundles (Zheng et al. 2019b). Besides the stakeholder requirements with adjustable parameters, for instance "change the default temperature of fridge from 8℃ to 5℃", lots of stakeholder requirements without well-predefined parameters are still hidden in the actual usage scenarios, which might not be realized by the users themselves as well. Hence, exploring the stakeholder requirements with context-awareness and adjusting the corresponding components in Smart PSS will significantly affect the final success of Smart PSS (Valencia Cardona et al. 2014).

From this perspective, one can find that requirement elicitation in Smart PSS not only copes with requirements themselves, but also the specific contexts in which the product-service solutions are carried out (Pacheco, García, and Reyes 2018). Although there are some studies applying semantic frameworks, such as ontologies or UML (Kim et al. 2009; McKay and Kundu 2014) to deal with requirements based on contexts, only one type of data (i.e. text) was employed. Owing to the smart and connected components (e.g. sensors and interactive user interfaces) in Smart PSS, huge-volume and heterogeneous data, including both textual data and sensor data can be collected and further analysed to extract useful requirements with better

annotations. Hence, it makes the requirement elicitation process in Smart PSS as a data-driven manner, where a new approach for requirement management is required to deal with various data sources. In addition, based on the heterogeneous real-time data collected which reflects the current status of usage contexts, it is possible to adjust the requirements not only at the early design stage but also the usage stage, making Smart PSS requirement management a closed-loop and ever-evolving ecosystem (Zheng, Wang, and Chen 2019).

Facing the aforementioned challenges, scarcely any study reports on a context-aware approach together with an appropriate information modelling technique in today's smart, connected environment. Aiming to fill this gap, a graph-based context-aware requirement elicitation approach in the Smart PSS is proposed in this work. It enhanced the existing graph embedding technique, i.e. Deepwalk, together with the pre-defined product/service/context ontologies to depict their in-context relations and further to discover prospective user needs. To better describe the proposed methodology, the rest of this paper is organized as follows. Section 2 gives an overview about the related works. Section 3 defines the research problem in a formalized way and explains how the requirements represented in the proposed method. Section 4 elaborates the proposed graph-based context-aware requirement elicitation approach. A case study on smart bike share requirement elicitation is further adopted to demonstrate its feasibility and effectiveness in Section 5. At last, in Section 6, main contributions and limitations of this work are concluded, and some future directions are also highlighted.

2 Related works

To clarify the importance of reliable data and information collection, and to further elicit implicit requirements with consideration of context information through them, related works of requirement elicitation in PSS are reviewed in this section.

2.1 Requirement elicitation approaches

Requirement elicitation (RE), as the first step of requirement engineering, has the most widespread concerns (Ambreen et al. 2018), which is to identify the reliable data sources, and then extract requirements from both explicit ones, e.g. voice of users, and implicit ones, e.g. sensor data (Hussain, Lockett, and Vasantha 2012). It has been widely adopted in the software development and product development fields in the past decades. The requirement elicitation methods are reviewed and discussed mainly from two aspects below.

From the perspective of requirement elicitation stage, most existing studies focus on extracting requirements in the early design stage rather than the usage stage. The conventional RE techniques, such as questionnaire, interview and focus group is usually time-consuming since it will take several weeks or even months for the operating teams to collect the voice of customers and analyse them. Those methods are still applicable in the early design stage but not suitable for the usage stage since the risks and requirements of Smart PSS are expected to be detected quickly and thereby actions are supposed to be taken in a short term. Some researchers have already realized the necessity of extracting requirements during lifecycle (Durugbo 2014) and the importance of using the real-time data in the usage stage (Hussain, Lockett, and Vasantha 2012). With the digitalization of physical products, e.g. via digital twin, and the mature of ICT techniques, huge volume and more kinds of data can be collected and at the same time products or PSS have the capability of reconfigure themselves (Abramovici et al. 2018), making it possible to extract requirements in usage stage and accordingly to improve the PSS. Lützenberger et al. (2016) realized the importance of exploiting information in usage stage. Product usage information is emphasized in their work for extracting product design requirements. Abramovici, Göbel, and Savarino (2017) addressed the feasibility of reconfiguration of smart products during usage phase based on virtual product twin. Besides the product usage information from product themselves, dynamic internet data including

product reviews, user attributes and products configurations are utilized for the analysis of requirements as well (Lai et al. 2019). Though the thought of applying data from usage phase has been raised up, the applications is still restricted to the functional adjustment (e.g. product reconfiguration and user demands on product features) of the smart products, the study of requirement elicitation on usage stage is still scarce.

From the aspect of the types of extracted requirements, explicit requirements (e.g. users' request) or well-defined requirements (e.g. predicted order quantity) are mostly studied by researchers (Salman et al. 2018; Murray, Agard, and Barajas 2018; Papanagnou and Matthews-Amune 2018; Misaghian and Motameni 2018). When dealing with explicit requirements among the crowdsourcing requirements gathering projects, natural language processing techniques or text-mining approaches are often applied. Salman et al. (2018) proposed an approach to automatically cluster functional requirements based on semantic measure. In their approach, functional requirements are represented as natural language and natural language processing process, including parsing requirements, tokenization, frequent tokens removal and so on, is applied simultaneously. Laurent and Huang (2009) utilized website for gathering and prioritizing requirements in a large-scale and distributed projects. Li et al. (2018) proposed an effective methodology to classify user requests by employing both project-specific and non-project-specific keywords and machine learning algorithms. Besides, big data analysis and machine learning methods are feasible to handle well-defined requirements since they are usually extracted from historical data (Murray, Agard, and Barajas 2018; Papanagnou and Matthews-Amune 2018). However, in Smart PSS, the implicit requirements without well-defined formats, for instance the needs users do not tell service providers or the ones they do not even realize, are usually omitted. Since they are hidden among user behaviours and usage scenarios (Shimomura et al. 2018), if we can explore them from the usage phase data, then the explored implicit requirements suffice to enhance PSS quality and customer satisfactions

(Tukker and Tischner 2017; Wang et al. 2019). Shimomura et al. (2018) proposed a scenario-based requirement elicitation method to extract keywords which constitute requirements from persona and scenarios although the data was historical and initial data without updates. Liu et al. (2017) have attempted to apply user behaviour data to extract requirements and thereby improve the performance of a software service. A requirement elicitation approach which study implicit requirements and intends to improve the performance of both products and services in Smart PSS remains to be further studied.

2.2 Requirement representation schema

Requirement representation schema, as the basis of requirement elicitation, is critical to decide which techniques and approaches suit for the requirement elicitation process among large scale and ever-evolving data, which can be further classified into four types, namely, *natural language*, *hierarchical structure*, *vector/matrix* or *tensor*, and *graph-based representation*.

Natural language refers to the human written or spoken language explicitly expressed by users. It is the most common and intuitive way for users to express their needs, some studies directly handle natural language for requirement extraction (Eyal Salman et al. 2018; Laurent and Cleland-Huang 2009). Though both human and machine have capability to handle natural language nowadays, those product-sensed data which has totally different format with natural language cannot be organized simultaneously. Thus, natural language has the drawback to organize heterogeneous types of data.

Hierarchical structure is a conventional requirement representation method. For instance, requirements abstraction model was proposed by Gorschek and Wohlin (2006) to decompose requirements into several level of abstraction to offer a continuous link from detailed requirements to the initial one. Geisberger et al. (2006) constructed a systematic requirement model called requirement engineering reference model to support the

interdisciplinary requirement exploration. Pohl and Sikora (2007) came up with a COSMOD-RE, an scenario- and goal-based architecture to support the co-design of requirements for the intensive software development. Berkovich et al. (2014) put forward a requirement data model to describe different types of requirements, including the product requirements and their relationships. Though hierarchical structure is intuitive for designer/engineers to manage the requirements, it is problematic to conduct large-scale computational analysis, which is necessary for requirement engineering in Smart PSS.

Vector, matrix or tensors are also a frequently used representation method to deal with requirements. Murray et al. (Murray, Agard, and Barajas 2018) analysed the customer transaction behaviours to discover the potential requirements. Hence, they converted the historical delivery transaction data as time-series vector format, by utilizing so-called dynamic time wrapping to provide better services. Papanagnou and Matthews-Amune (2018) also studied the customer transaction behaviours with the consideration of both ERP data and Internet information. Wang et al (2017) formalized user requirements with several requirement indices as requirement matrix, where requirements are represented as rows, and compared with product indices, so as to build up them from low level to high level. Misaghian and Motameni (2018) adopted tensors which is composed of functional requirements, non-functional requirements and stakeholders' preferences. Though vector, matrix and tensors are flexible for various computational methods, such as deep-learning algorithms, most researchers either restrict the scope of requirements as well-defined requirements (e.g. predicting future demands) based on historical data or discuss the importance of the stakeholders' preference, while omitting the context of requirements.

Graph is an intuitive way to represent complex system which is usually used in PSS. Chen and Occeña (2000) developed an expert system to organize requirement information by employing a graph decomposition algorithm which is adapted from Owen's algorithm. Kim

and Suzuki (2015) introduced a graph-based social context representation schema by employing the interactions between service providers, service receivers and physical touchpoints to retrieve similar cases and exploit related know-hows for further design strategies. Kim et al. (2009) also proposed PSS representation schema via graphs to organize diverse requirements and providing a case study of meal assembly kitchen. They all used graph to explore hidden relationships between various components, rather than the potential requirements lying behind those components. Based on the strengths of graph, such as good performance for representing complex systems, suitable for organizing evolving relationships and easy for maintenance, it is a good representation schema to mine the implicit requirements in Smart PSS. However, scarcely any studies adapt graph-based representation for requirement elicitation task in PSS.

To summarize, many previous studies had studied RE approaches from the lifecycle perspective to the operational perspective (as shown in Table 1), but few ones have holistically considered requirements, especially implicit requirements, based on heterogeneous data together with context information in both the early design stage and usage stage of Smart PSS. With the support of ICT techniques, the critical phases of eliciting requirements can be prolonged from only early design stage to usage stage in order to further evolve Smart PSS design itself. Meanwhile, the research scope is also supposed to be expanded to discover potential implicit requirements in connection with contexts. In order to conform the heterogeneous data with various data types to a unified RE method, a general data model containing context information should be put forward to represent the key information in requirements. Except for the theoretical breakthrough in RE of Smart PSS, most existing studies had only investigated the practical applications in product-related fields (e.g. smartphone design) or service-related fields (e.g. mobile app design). The PSS applications which take both context and products/services into a holistic consideration remain not well-investigated.

Table 1. Comparison among existing requirement elicitation methods from literature

Reference	Theoretical consideration					Application consideration
	Comparable RE methods	**Lifecycle consideration (i.e. on which stage)**	**System perspective (i.e. the type of requirements)**	**Operational consideration (data sources)**	**Requirements with context-aware or not**	**Case study**
(Li et al. 2018)	Crowdsourcing and topic analysis	Early design stage	Explicit requirements	Initial information (i.e. explicit user requests)	No	Services (software projects)
(Shimomura et al. 2018)	Topic analysis based on persona and scenarios	Early design stage	Explicit requirements	Initial information (i.e. predefined usage scenarios)	Yes	PSS (urban development)
(Lai et al. 2019)	Online data mining	Usage stage	Both explicit requirements and implicit requirements	Initial information (i.e. product configuration, user attributes) and usage data (i.e. user reviews)	No	Product-related（smartphone design）
(Murray, Agard, and Barajas 2018)	Data mining techniques to extract well-defined requirements	Usage stage	Explicit requirements	Usage data (i.e. historical transaction data)	No	Product-related（delivery quantity prediction）
(Liu et al. 2017)	Cybernetics-based approach	Usage stage	Implicit requirements	Usage data (i.e. user behaviour data)	Yes	Services (software apps)

3 Problem description and model formulation

3.1 Problem description

To fill the abovementioned research gaps in Section 2, this study intends to extract existing requirements or implicit requirements based on the usage contexts. Based on a well-known requirement boilerplate, namely Rupp's boilerplate (Arora et al. 2014), a requirement can be represented as "Under what context, system component(s) shall/should/will do process", shown as **Figure 1**.

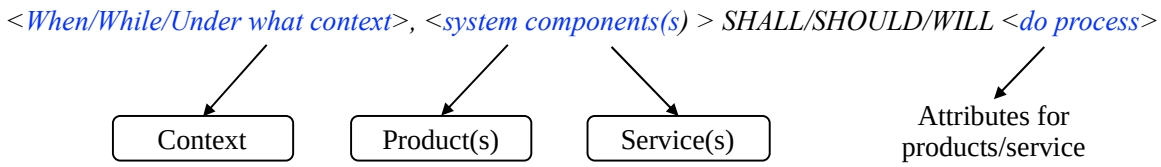

Figure 1. Requirement representation and its relations with Smart PSS components

Here the *context* refers to the environmental factors which can affect the performance of the product-service solutions, such as temperature and location. *System components* means either product components or service components in Smart PSS. *Process* is the actions that the product/service components can offer, which can be regarded as attributes belonging to the product/service components. For example, a requirement which expressed as 'Under CPU's temperature is high, the notebook thermal module (computer radiator) should work' can be decomposed as two entities, i.e. 'CPU's temperature' and 'notebook thermal module'. In this way, totally three kinds of entities constitute the key skeleton of requirement, namely context, product and service. At the same time the interactions between them are able to reflect the information among the requirements. By considering the

connections between requirements and components in Smart PSS, eliciting requirements is to explore the interactions between context(s), product(s) and service(s).

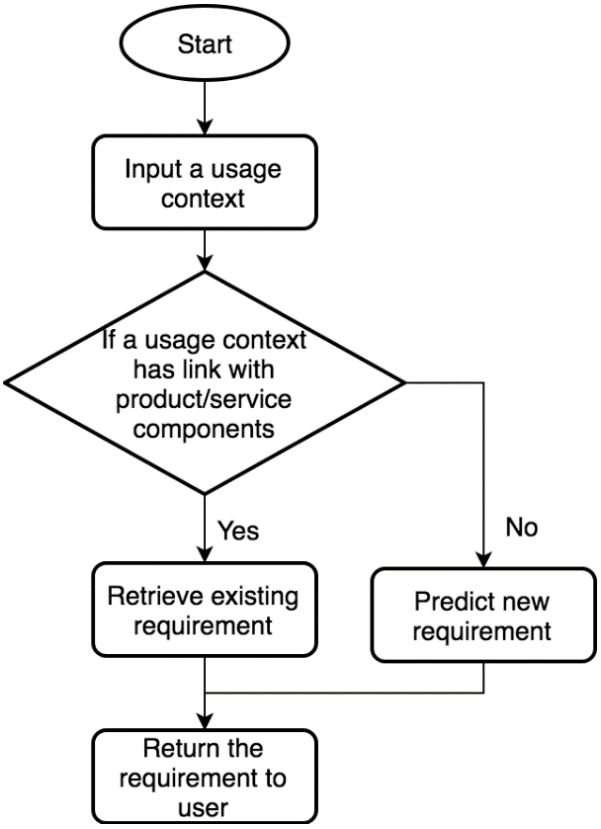

Figure 2. The working flow of the proposed approach

From this perspective, an overall working flow of the proposed requirement elicitation approach can be seen in **Figure 2**. Given a context, if there is any historical requirement containing this context and its links to other products/services, then retrieve them as a requirement. If it is a new context or there is no links attached to it, then we need to discover the most relevant products/services among the product family and the service pool, treating them as a new requirement. Hence, facing the situation of new context or no links with the given context, the requirement elicitation model can be defined as a link prediction problem between the context and the other products/services. Furthermore, the

link prediction problem can be transferred as a ranking problem among all the potential products and services.

3.2 Model formulation

In the proposed approach, a requirement graph is built up. The context, products and services are treated as nodes in a requirement graph (RG) and their interactions are represented as the links between the nodes. The RG can be represented as $RG = < V, E >$ where the vertex set $V = P \cup S \cup C$ is the union of three types of nodes: product components nodes, service components nodes and context nodes. Edge set E consist of five subsets CP, CS, PP, SS and CC. The nodes and the pairwise or group relationships are specified from the information collected in Smart PSS as follows.

3.2.1 Types of nodes

All the nodes are generated from the information from service provider, including product family structure, service pool and knowledge from domain experts, which is pre-processed as domain ontologies.

- **Products components (*P*)** refers to the components or modules which constitute the physical system which interact with users in Smart PSS. The information of product components is organized in a product ontology and represented in a tree diagram, where component p_i refers to the *i*th components or modules.
- **Services components (*S*)** means the existing services that the service providers can offer to users during the usage stage. Similarly, the information of service components is also stored in a service ontology as a tree diagram, where s_j is the jth service in service ontology. To the authors' knowledge, there is still lack of a

comprehensive consensus about what services can be provided in PSS, let alone to the services in Smart PSS. Hence, based on the studies from (Abramovici et al. 2009), a clear scope of service in Smart PSS is clarified, as shown in **Figure 3**. It mainly consists of two types of services, namely *digitalized services* and *e-services*. *Digitalized services* refer to the services which is highly dependent on the physical products, as an integrated hardware and software bundle such as digital twin of machine tool, while *e-services* refer to the software-based services, which are independent with the physical products, weather forecasting app in smart phone for example (Zheng, Chen, and Shang 2019).

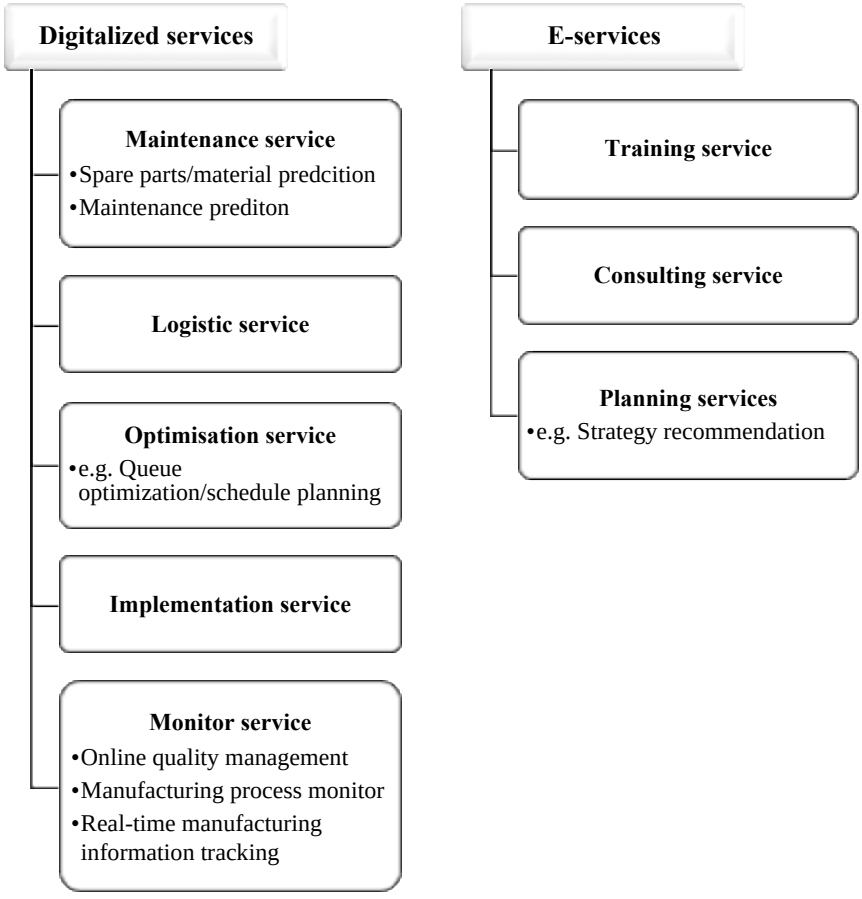

Figure 3. Types of services in the Smart PSS

- **Usage contexts ($\boldsymbol{C}$)** is the relevant factors which cause various performance while using the product-service solution. For example, temperature and the wind speed are two factors which affect the user experience of riding a bike, so they should be considered while extracting requirements. Besides environment factors, some user behavioural-based data, for instance riding distance, should also be included into the usage contexts. A usage context is expressed as c_r, indicating that it is the rth context in the context ontology.

3.2.2 Co-occurrence relationships between nodes (edges)

Moreover, based on the aforementioned requirement compositions, the possible relationships are denoted as the edges in-between them, and can be further classified into five forms, respectively as shown in **Figure 4**. In this approach, the co-occurrence relations can be extracted if two nodes appear simultaneously in a user comment.

Product-Product relationship (PP). If two product components co-occurred in users' comments, they have the *PP* between them. For example, bike saddle and the seat clamp appear together in a user comment, then a ***PP*** relationship connects the product node 'saddle' and the product node 'seat clamp'.

Service-Service relationship (SS). In users' comments, various services can be suggested under the same context, hence ***SS*** should be considered between two service elements to present the co-occurrence between them. As an example, under the scenario of bike theft, 'install camera' or 'install bike dock with locks' are two different but similar services. A SS relationship should be added between service node 'install camera' and the service node 'install bike dock with locks'.

Context-Context relationship (CC). In reality, some product-service bundles can suit for various contexts, hence it is possible that different context appearing in same user comment. In order to imitate this phenomenon, the *CC* between contexts should be specified in the requirement graph.

Product-Context relationship (PC). If a product mentioned when users descripting the context, then *PC* exists in the graph. If product node 'saddle' and context node 'high humidity' are mentioned together by users, then a ***PC*** relationship exist between them.

Context-Service relationship (CS). If related services are suggested with the context, then *CS* exist. Similarly, if service node 'change saddle' are suggested by users with the context node 'high humidity', then they are connected by ***CS*** relationship.

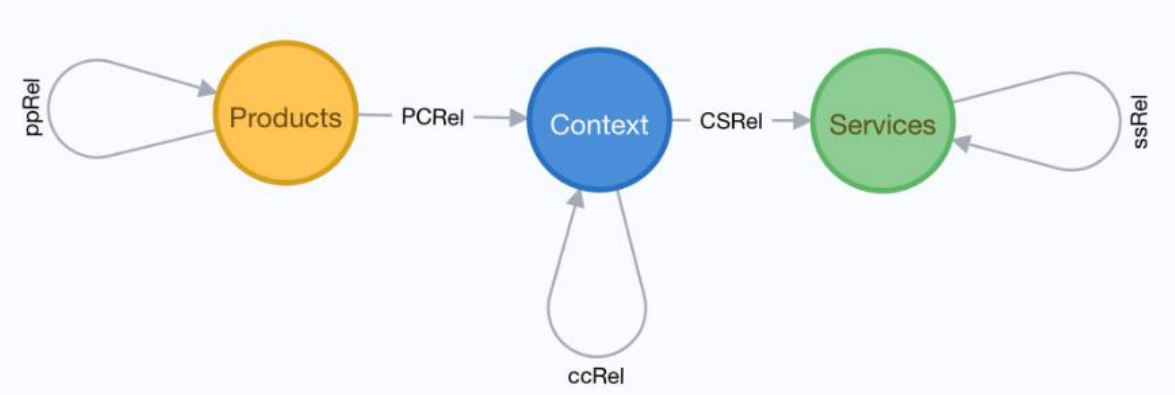

Figure 4. Schema of five different relationships between nodes

In this way, a requirement graph with three types of nodes and five types of edges can be built up, as shown in **Figure 4**. The requirements in the form of sentences are treated as the edges which contain heterogeneous nodes in RG, denoted as $< p_i, s_j, c_r >$ in a mathematical way.

4 Proposed graph-based PSS requirement elicitation approach

Since the core of the requirement elicitation approach intends to predict the most relevant

products/services based on the usage context, which is specified as a link prediction task in the former section, we proposed an enhanced Deepwalk-based approach to elicit requirement in this section. It is a continuous work based on our previous contribution (Z. Wang et al. 2019). The overall flowchart of the requirement elicitation approach is shown as **Figure 5**.

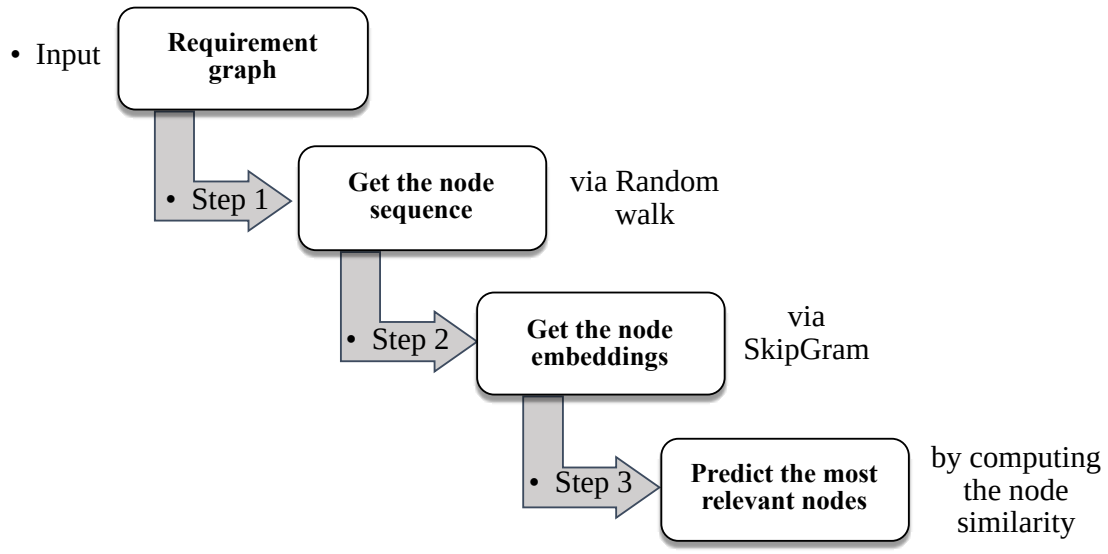

Figure 5. The overall flowchart of the proposed requirement elicitation approach

4.1 Input: Requirement graph

Based on the model aforementioned, the RG is represented as $RG = < V, E >$ where the vertex set $V = P \cup S \cup C$ is the union of product components, service components and context components. In order to represent the finite RG in a mathematical manner, an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{|V|\times|V|}$ is defined to represent it, in which each element is:

$$a_{i,j} = \begin{cases} 1, \ if \ v_i \ and \ v_j \ have \ edge \\ 0, \ else \end{cases} \quad (1)$$

The adjacency matrix $\boldsymbol{A}$ is a diagonal matrix where its diagonal elements are zero. The three types of nodes, i.e. P, S and C, are all denoted as one-hot encoding using binary code

{0,1} to represent elements with dimension $|V|$. If there is a large-scale requirement graph with millions of nodes, then the dimension $|V|$ will also be millions, which is significantly high. At the same time, the vertices' matrix $V \in \mathbb{R}^{|V|\times|V|}$ is a sparse matrix in which only one element in a vector is 1 and all other elements are 0. Furthermore, by using one-hot encoding, the distance of any two nodes are same all the time, which cannot show their proximity in RG.

To avoid these defects, dimension reduction is required for the sake of computation efficiency (Cai, Zheng, and Chang 2018) and we intend to represent nodes as low-dimensional embeddings. The distance between two 'embeddings' will be closer if they have similar implications. These advantages make embeddings a good format to perform machine learning tasks (Jin et al. 2016), for instance link prediction. Motivated by this, in this work, the high-dimensional one-hot encoding nodes are transformed into low-dimensional embeddings, and further analysis is performed on these embeddings subsequently.

4.2 Step 1: Get the sequence order via Random Walk

Deepwalk (Perozzi, Al-Rfou, and Skiena 2014) is one of the graph embedding methods for learning latent representations of the nodes in a graph. The Deepwalk algorithm consists of two main components, namely *random walk generator* and *Skipgram* (Mikolov et al. 2013). It uses truncated random walks to generate node sequences and treats the sequences as the equivalent of sentences. A random walk is denoted as a vector which is rooted at vertex v_i within a window w, i.e. $W_{v_i} = \{v_{i-w}, v_{i-w+1}, \dots, v_{i-w} - 1, v_{i+w},\}$. The chosen of the variant nodes in a walk is a stochastic process. **Table 2** shows the pseudo code of generating a random walk in a requirement graph.

Table 2.The pseudo code of a random walk generation procedure

The algorithm architecture of a random walk generation procedure	
Input: walk_length l, start_node v_i, requirement graph RG **Output:** a random walk W	
	Initialisation:
1:	Given adjacency matrix A to represent requirement graph RG
2:	Given walk_length l
3:	Given start node v_i
4:	Set an empty list Was a random walk
5:	Set l' as the current walk length in the random walk W
	Generate the random walk:
6:	**WHILE** $l' < l$ **do** :
7:	**IF** no. of current_node_neighbor >0 **do**:
8:	Append a random node among current_node_neighbor to the walk
9:	**ELSE** break
10:	**END IF**
11:	**END WHILE**
12:	**RETURN** W
	END

4.3 Step 2: Get the node embeddings via SkipGram

In terms of the node sequences W generated via random walk, we aim to learn the latent semantic embeddings for each node in requirement graph G. SkipGram (Mikolov et al. 2013) as a language model is adopted in this subsection to learn the mapping function between the nodes and their embeddings. A mapping function $\Phi: v \in V \longmapsto \mathbb{R}^{|V|\times d}$ is introduced, where $d \ll |V|$. It intends to maximize the co-occurrence probability among the nodes that appear within a window w, i.e. maximize the co-occurrence probability $Pr\left(\{v_{i-w}, \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+w}|v_i\}\right)$ of the nodes when given one node to predict both the right and left nodes of the given one. This is what the requirement elicitation approach intends to solve, namely, to predict the relevant products/services based on the given context node where SkipGram is a suitable model to derive the embeddings of nodes. The task of

using one node to predict the nodes on both sides can be formulated as an optimization problem with the objective function as below:

$$\min_{\Phi} -log\ Pr(\{v_{i-w},\ \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+w}|v_i\}) \quad (2)$$

Considering the embeddings of the nodes, the objective function can be denoted as:

$$\min_{\Phi} -log\ Pr(\{v_{i-w},\ \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+w}|\Phi(v_i)\}), \quad (3)$$

, where $\Phi(v_i)$ is the embedding of vertex v_i. Furthermore, the SkipGram model approximates the co-occurrence probability in Equation (3) using an independence assumption as:

$$Pr(\{v_{i-w},\ \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+w}|\Phi(v_i)\} = \prod_{j=i-w, i\neq j}^{i+w} \Pr(v_j|\Phi(v_i)) \quad (4)$$

For the sake of computational efficiency, hierarchical Softmax and stochastic gradient descent (SGD) are utilized to approximate the probability distribution and to optimize the parameters, respectively.

4.4 Step 3: Predict the most relevant nodes

After getting the node sequences via random walk and learning the embeddings $\Phi(v_i) \in \mathbb{R}^d$ of the nodes via SkipGram model, we can transfer the task of computing the similarity of nodes as computing the distance between the embeddings of the corresponding nodes. In order to simplify the algorithm complexity, cosine similarity is adopted to compute the proximity of the nodes, which is shown as follow.

$$S(v_i, v_j) = \cos(v_i, v_j) = \frac{\Phi(v_i)\cdot\Phi(v_j)}{\|\Phi(v_i)\|\cdot\|\Phi(v_j)\|} \quad (5)$$

In the context of Smart PSS, the optimized requirement should contain both products and services based on the given context. Therefore at least three nodes should be included in a requirement, composing a tuple <P, C, S>. Since both the similarities s_{PC} and s_{CS}will be calculated, a total similarity s_t score should be set up as well, which is denoted as:

$$s_t = s_{PC} \times s_{CS} \tag{6}$$

The pseudo code of the link prediction including the aforementioned similarity computation is listed in **Table 3**. Tuples with related nodes and top 10 similarities are extracted. Tuples go through the direct-connected nodes and indirect-connected nodes with the given initial nodes, embracing implicit relationships which may not be expressed by customers. By leveraging the approach described above, when a node among the requirement graph was triggered by high frequency, its related nodes with high co-occurrence relations are also extracted. Those derived nodes and the original given node are connected with either direct or indirect edges, and their combination constitutes the explicit or implicit requirements.

Table 3. The pseudo code of extracting similarities between nodes

The algorithm architecture of extracting similarities between nodes	
Input: id of the start node, adjacency matrix A of the requirement graph, vertex set V and the embeddings set Φ of nodes **Output:** rank of similarity between nodes	
	Initialisation:
1:	Given adjacency matrix A to represent requirement graph RG
2:	Given the id of the start node v_0, set the id as id_0
3:	Given the embeddings set Φ of nodes
4:	Set an empty list S as the rank of similarity between nodes
	Compute the similarity between nodes:
5:	Get the embedding Φ_o of the start node
6:	**For** product node p_i in vertex set V **do**:

7: Get the embedding Φ_{p_i} of the product node p_i

8: Compute the similarity between the start context node and the current product node $s_{PC} = \frac{\Phi_o \cdot \Phi_{p_i}}{\|\Phi_o\| \cdot \left\|\Phi_{p_i}\right\|}$

9: **For** service node s_j in vertex set V **do**:

10: Get the embedding Φ_{s_j} of the service node s_j

11: Compute the similarity between the start context node and the current service

12: node $s_{CS} = \frac{\Phi_o \cdot \Phi_{s_j}}{\|\Phi_o\| \cdot \left\|\Phi_{s_j}\right\|}$

13: Compute the total similarity based on the current product node and service node $s_{t_{ij}} = s_{PC} \times s_{CS}$

14: Append the current $s_{t_{ij}}$ to the list S

15: **End for**

16: **End for**

Rank the similarity score:

17: Rank the elements in list S from high to low

18: **RETURN** rank of similarity between nodes

END

5 Experiment and results

To empirically demonstrate the proposed approach, an illustrative example of smart bike design improvement is adopted to demonstrate how the proposed method works empirically. It is aimed to explore the stakeholder requirements which contains relative components (i.e. products and/or services) and the specific contexts in a smart bike system.

5.1 Dataset preparation

The data comes from a crowdsourcing activity of review collection about bike ridings in Melbourne with the objective of drafting Bicycle Plan 2016-2020, and the raw data can be found online via https://data.melbourne.vic.gov.au/Transport-Movement/Public-comments-on-2016-20-Bicycle-Plan/8kn4-yjni. Totally 1350 comments, 1000 individual spots and 4500 supports were collected from the public. To simplify the problem and reduce

the computation time, only the first 100 comments are selected for analysis. Among the 100 comments, 117 pieces of information are extracted, and they are treated as nodes. Their properties such as ID, content and category are stored as a .csv dictionary for retrieval and are partially listed in **Table 4**. Only 108 nodes with the aforementioned co-occurrence relationships are used for requirement extraction.

Table 4. A partial list of smart bike node properties

id(n)	content	Category
0	bike	Products
1	car	Products
2	parked cars	Products
3	drinking fountains	Products
4	sensors on traffic light	Products
5	tree	Products
…	…	…
40	foot traffic is high	Context
41	too many overtaking	Context
42	heavy bike traffic	Context
43	traffic congestion at peak time	Context
44	traffic congestion at intersection	Context
45	traffic congestion at major routes	Context
…	…	…
109	add clear guide sign beside roads	Services
110	build underpass	Services
111	raise pavement to footpath level	Services
112	rearrange bike traffic light	Services
…	…	…

Except for the node dataset, edge datasets including P-C, C-S, P-P, C-C and S-S are organized as well, as shown in **Table 5**. They are extracted from crowdsourcing public comments for the Bicycle Plan 2016-2020, where edges are built in accordance with the node IDs.

Table 5. A partial list of smart bike edges

Product	**Source_id**	**Context**	**Target _id**
bike	0	cars block bikes	52
car	1	cars block bikes	52
parked cars	2	bike lanes zig zag	66
drinking fountains	3	thirsty cyclists	69
sensors on traffic light	4	waste of time for cyclists	50
camera	7	bike theft	68
dock	8	no adequate space for both cars and bikes	54
dock	8	no greenery	61
dock	8	bike theft	68
…	…	…	…
Context	**Source_id**	**Service**	**Target_id**
bike theft	68	add cameras near bike docks	100
bike theft	68	add docks	101
thirsty cyclists	69	build drinking fountains	102
no clear bike lane signage	65	build separate bike lanes	103
tram stop	71	build new bike lane	104
park	73	build new bike lane	104
…	…	…	…
Product	**Source_id**	**Product**	**Target_id**
bike	0	ramp	26
bike	0	car	1
car	1	bike	0
car	1	ramp	26
sensors on traffic light	4	bike traffic light	17
tree	5	shrub	6
tree	5	dock	8
…	…	…	…
Context	**Source_id**	**Context**	**Target_id**
heavy bike traffic	42	too many overtaking	41
too many overtaking	41	heavy bike traffic	42
no clear bike lane signage	65	traffic congestion at intersection	44
need to press the button to activate the traffic light for cyclists	82	no clear signage that if the traffic light is activated or not	47
need to press the button to activate the traffic light for cyclists	49	no clear signage that if the traffic light is activated or not	47
…	…	…	…
Service	**Source_id**	**Service**	**Target_id**
add docks	35	add cameras near bike docks	34
add cameras near bike docks	34	add docks	35
add clear signage on bike lane	106	build separate bike lanes	37
widen bike lane	119	build separate bike lanes	37
widen bike lane	119	add clear signage at the beginning of bike lane	104
…	…	…	…

5.2 Construction of a requirement graph

The requirement graph was constructed in terms of the edge lists, completed through a graph database, Neo4j. A total of 119 nodes constitute the graph including 108 ones with edges are shown in **Figure 6**. In the graph, yellow nodes with contents stands for the product nodes. Similarly, green nodes and blue nodes with content refer to service nodes and context nodes, respectively. Initially, 119 nodes are all represented with one-hot encoding in $\mathbb{R}^{119}$. Among the 119-dimensional vectors, only positions which have edges with other nodes will be represented as '1' and the rest of positions without edges are represented as '0', becoming sparse vectors.

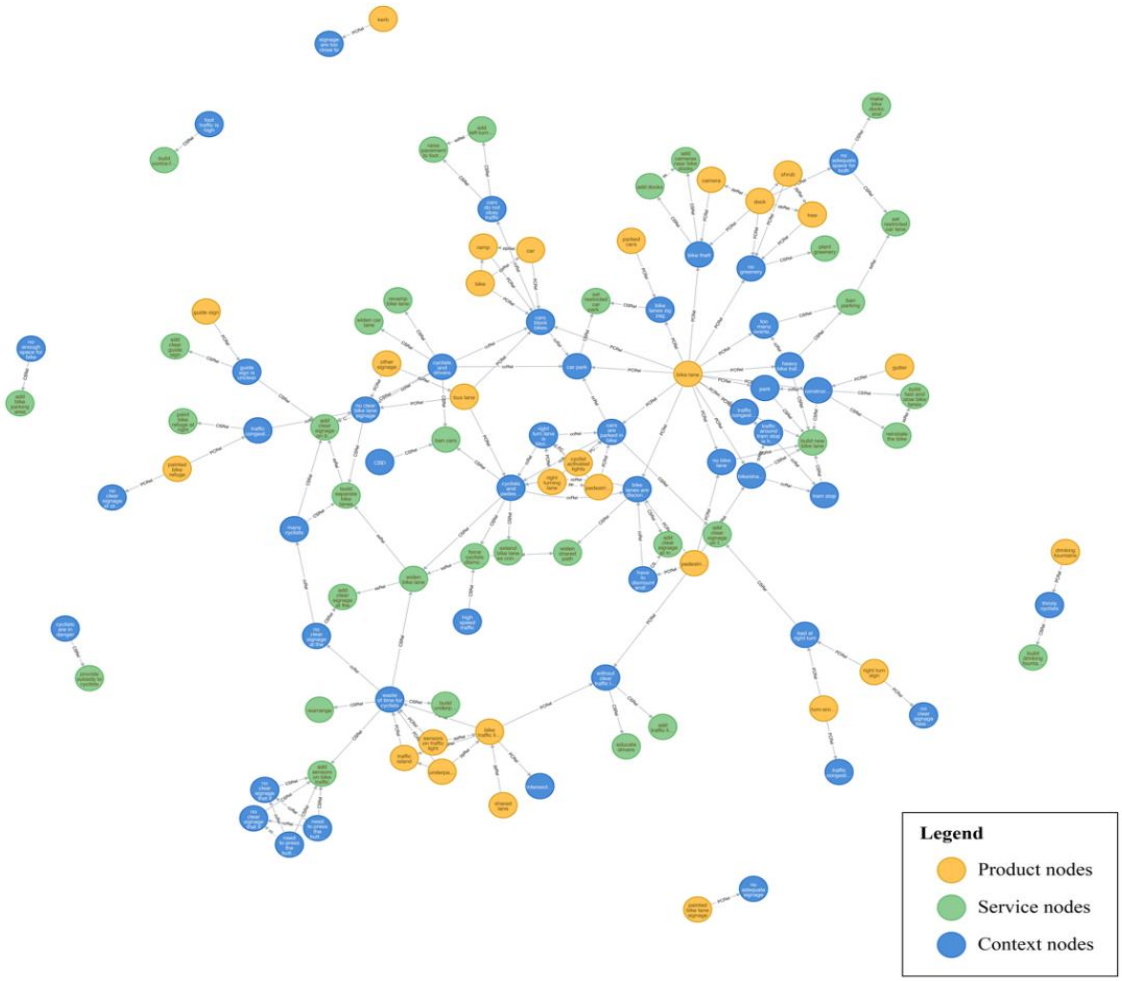

Figure 6. Requirement graph of Smart Bike Ridings

5.3 Requirement elicitation process

The goal of this phase is to project the nodes into a relatively low-dimensional space and then extract the similarity between nodes. By leveraging Deepwalk, the 108 nodes with edges are embedded into a 16-dimensional space $\mathbb{R}^{16}$. The learned representation in $\mathbb{R}^{16}$ can be easily conducted by computational operation. As already mentioned, cosine similarity between two nodes are computed to represent the correlation of nodes. The codes are completed by Python on Jupyter Notebook. Given the id of initial context node, the approach is able to return the other two most relevant nodes as tuples. Tuples are ordered by similarity high to low. Serval results are shown in **Table 6**.

For example, in **Table 6**, from the first tuple which begins with context node 'Traffic congestion at intersection(c)', the two most relevant nodes are 'paint bike refuge at right place(s)' and 'painted bike refuge(p)' with a similarity value of 0.98, indicating that no painted bike refuge might cause the traffic congestion at intersections. And the requirement extracted from this tuple can be expressed as '*When the traffic congestion at intersection, bike refuge should be painted at right place*'. Similarity, other tuples can be recovered as requirements as well, serving as reference for designers/engineers to make further design improvements in Smart PSS.

In summary, the extracted tuples reveal the direct relationship and indirect relationship between nodes, which effectively assist the designers/engineers to find the problem hidden behind the large volume and heterogenous context/product/service information.

Table 6. Part of extracted tuples

Initial context nodes	Product/service node	Product/service node	Similarity
Traffic congestion at intersection(c)	paint bike refuge at right place(s)	bike refuge(p)	0.98
no greenery(c)	shrub(p)	plant greenery(s)	0.978
construction blocks bike lane(c)	gutter(p)	reinstate the bike lane once the construction is complete(s)	0.974
	gutter(p)	build new bike lane(s)	0.916
waste of time for cyclists(c)	rearrange bike traffic light(s)	sensors on traffic light(p)	0.97
	rearrange bike traffic light(s)	traffic island(p)	0.967
	build underpass(s)	sensors on traffic light(p)	0.956
cyclists and drivers use same lanes(c)	revamp bike lane(s)	bus lane(p)	0.964
	widen car lane(s)	bus lane(p)	0.948
no clear bike lane signage(c)	bus lane(p)	revamp bike lane(s)	0.932
	paint bike refuge at right place(s)	painted bike refuge(p)	0.91
bike theft(c)	add docks(s)	camera(p)	0.94
	add cameras near bike docks(s)	camera(p)	0.928
Bad at right turn(c)	right turn sign(p)	add clear signage on traffic light(s)	0.945
no adequate space for both cars and bikes(c)	make bike docks end to end(s)	dock(p)	0.939
CBD(c)	ban cars(s)	bus lane(p)	0.91

(‘c’ = context, ‘p’ = product, ‘s’=service)

5.4 Discussion

From the experiment result of the smart bike-sharing system example, it reveals that the proposed graph-based RE approach can achieve the effective elicitation of user's implicit requirements in the specific context, with co-related product and/or services components triggered accordingly. Hence, user behavior can be readily recognized and leveraged to the redesign of engineering products and its generated services with context-awareness. Meanwhile, compared to the existing studies of which requirements are expressed by natural language with pre-defined template, the proposed context-product-service data model is novel and effective to organize the requirements in a data-driven manner. This result was proved by the successful transformation between the natural language requirements and the <P, C, S> tuples without losing key information. Furthermore, by using the algorithm of obtaining the nodes embeddings, the proximity between nodes and the reasonable meanings of requirements are both retained, as demonstrated by the results with high similarity in Table 6. Owing to the advantages of the proposed generic RE methodology, it can be readily extended to many other applications in the Smart PSS field, such as smart water dispenser maintenance service for smart living (Zheng et al. 2019), and automatic engineering product-service change management in smart manufacturing (Zheng, Chen, Wang 2019), to name a few.

Nevertheless, in order to adopt the proposed graph-based RE approach in practice, several managerial insights should be addressed as well. Firstly, at the operational level, managers/operation teams are expected to apply the graph databases to store and operate the heterogeneous data to fit the proposed requirement graph and the context-product-service data model. It requires the transformation from relational database (SQL-based ones) to the NoSQL database. Secondly, at the product-service design level, a context-awareness module which can process contextual information and judge the states of usage scenarios is supposed to be

embedded into the smart products as well. This can be achieved by the embedded AI of the microcontroller, and regarded as one of the key components apart from the hardware and software components. Moreover, at the system infrastructure (technology stack) level, due to the exponential increase of ever smarter devices in the future, a cloud-edge computing platform will be the ideal basis for the proposed approach to handle the large-volume IoT data, avoiding the large consumption of computation resources, energy and bandwidth with high response time (Zheng, Wang, and Chen 2019). Last but not least, management actions including user incentive mechanisms, which can enhance the level of context-awareness with stakeholders' contributed reliable data sources should also be considered.

6 Conclusion

Smart PSS are fairly complex and ever-evolving eco-systems with the aim of satisfying individual customer's requirements. By embracing the cutting-edge ICTs, large-scale context data become more readily accessed by companies via hardware sensing and crowdsourcing channels with both human and machine intelligence. Companies can leverage on them to extract large amount of useful information, especially the user requirements, to facilitate their solution design. Hence, a proper requirement elicitation approach considering context data in the usage stage is required to assist the service providers to find out the implicit requirements which has potential to improve the performance of their product-service bundles, so that the companies can preserve their competitiveness and high profits in today's fierce market. As an exploratory study, this paper proposed a novel graph-based context-aware requirement elicitation method in the Smart PSS, and further discussed the relationships between products, usage conditions and services. The main contributions of this work can be summarized into three aspects:

A novel graph-based requirement representation manner was defined. In this paper, a generic requirement graph schema was introduced, where requirements are treated as sentences,

while key information is regarded as nodes or words in sentences. Hence, by merging all three types of nodes and five co-occurrence relations, the requirement elicitation task is standardized as the problem of mining the co-occurrence relations between key information of requirements with computation efficiency.

A Context-Product-Service (CPS) data model which can represent the key information of the requirements was proposed. It is a general data model which can conform to the heterogeneous data with various data types. By applying the proposed data model, the requirements in Smart PSS can be enriched by not only textual data but also context data composed of multiple sensor data, making the context-awareness possible while extracting requirements in Smart PSS.

A systematic context-aware requirement elicitation method was proposed. Requirements can be extracted from the direct and indirect relationships between nodes based on their high co-occurrence probability. Requirements can be extracted with higher reliability by integrating context information, which aims to recur the problem scenarios into the task.

Moreover, an illustrative example of requirement elicitation for smart bike-share system is further utilized to validate the feasibility and effectiveness of the proposed graph-based requirement elicitation approach. Despite these achievements, this research still has some limitations. For instances, this proposed approach only considers the structure of graph, while the edge weights also embrace lots of information (e.g. strength of the co-occurrence relationship). Hence, it should be considered in the graph, as the input of requirement elicitation model as well. Furthermore, factors dramatically affect the quality of Smart PSS (e.g. safety and dynamics of requirements) should also be involved in the graph. Nevertheless, the authors hope this paper can be seen as the foundation to clarify the task of requirement elicitation in the Smart PSS context. Meanwhile, some future research works can be further explored to 1) use

ontology learning techniques or knowledge graph to introduce external concepts (including product concepts, service concepts and context concepts) into the existing requirement graph in order to evolve it in a long term, and 2) apply the extracted stakeholder requirements to assist stakeholders for decision makings (e.g. reconfigure the functions of the product-service bundles) in the usage phase in order to achieve value co-creation of Smart PSS with the participation of users.

Acknowledgement

The authors wish to acknowledge the financial support from the National Research Foundation (NRF) Singapore and Delta Electronics International (Singapore) Pte Ltd., under the Corporate Laboratory@ University Scheme (Ref. SCO-RP1; RCA-16/434) at Nanyang Technological University, Singapore.

References

Abramovici, M, M Neubach, M Schulze, and C Spura. 2009. "Metadata Reference Model for IPS2 Lifecycle Management." In . Cranfield University Press.

Abramovici, Michael, Jens Christian Göbel, and Philipp Savarino. 2017. "Reconfiguration of Smart Products during Their Use Phase Based on Virtual Product Twins." *CIRP Annals* 66 (1): 165–168.

Abramovici, Michael, Philipp Savarino, Jens Christian Göbel, Stefan Adwernat, and Philip Gebus. 2018. "Systematization of Virtual Product Twin Models in the Context of Smart Product Reconfiguration during the Product Use Phase." *Procedia CIRP* 69 (1): 734–739.

Ambreen, Talat, Naveed Ikram, Muhammad Usman, and Mahmood Niazi. 2018. "Empirical Research in Requirements Engineering: Trends and Opportunities." *Requirements Engineering* 23 (1): 63–95.

Arora, Chetan, Mehrdad Sabetzadeh, Lionel C Briand, and Frank Zimmer. 2014. "Requirement Boilerplates: Transition from Manually-Enforced to Automatically-Verifiable Natural Language Patterns." In , 1–8. IEEE.

Berkovich, Marina, Jan Marco Leimeister, Axel Hoffmann, and Helmut Krcmar. 2014. "A Requirements Data Model for Product Service Systems." *Requirements Engineering* 19 (2): 161–186.

Cai, Hongyun, Vincent W Zheng, and Kevin Chang. 2018. "A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications." *IEEE Transactions on Knowledge and Data Engineering*.

Chen, Chun-Hsien, and Luis G Occeña. 2000. “Knowledge Decomposition for a Product Design Blackboard Expert System.” *Artificial Intelligence in Engineering* 14 (1): 71–82.

Durugbo, Christopher. 2014. “Strategic Framework for Industrial Product-Service Co-Design: Findings from the Microsystems Industry.” *International Journal of Production Research* 52 (10): 2881–2900. doi:10.1080/00207543.2013.857054.

Eyal Salman, Hamzeh, Mustafa Hammad, Abdelhak-Djamel Seriai, and Ahed Al-Sbou. 2018. “Semantic Clustering of Functional Requirements Using Agglomerative Hierarchical Clustering.” *Information* 9 (9): 222.

Geisberger, Eva, Manfred Broy, Juergen Kazmeier, Daniel Paulish, and Arnold Rudorfer. 2006. “Requirements Engineering Reference Model (REM).”

Gorschek, Tony, and Claes Wohlin. 2006. “Requirements Abstraction Model.” *Requirements Engineering* 11 (1): 79–101.

Grubic, Tonci, and Ian Jennions. 2018. “Remote Monitoring Technology and Servitised Strategies – Factors Characterising the Organisational Application.” *International Journal of Production Research* 56 (6): 2133–2149. doi:10.1080/00207543.2017.1332791.

Hussain, Romana, Helen Lockett, and Gokula Vijaykumar Annamalai Vasantha. 2012. “A Framework to Inform PSS Conceptual Design by Using System-in-Use Data.” *Computers in Industry* 63 (4): 319–327.

Jin, Zhipeng, Ruoran Liu, Qiudan Li, Daniel D Zeng, YongCheng Zhan, and Lei Wang. 2016. “Predicting User’s Multi-Interests with Network Embedding in Health-Related Topics.” In , 2568–2575. IEEE.

Kim, Yongse, and Kumiko Suzuki. 2015. “Social Context Representation in Product-Service Systems with Internet of Things.” *Open Journal of Social Sciences* 3 (07): 187.

Kim, YS, E Wang, SW Lee, and YC Cho. 2009. “A Product-Service System Representation and Its Application in a Concept Design Scenario.” In . Cranfield University Press.

Lai, Xinjun, Qixiang Zhang, Qingxin Chen, Yunbao Huang, Ning Mao, and Jianjun Liu. 2019. “The Analytics of Product-Design Requirements Using Dynamic Internet Data: Application to Chinese Smartphone Market.” *International Journal of Production Research* 57 (18): 5660–5684.

Laurent, Paula, and Jane Cleland-Huang. 2009. “Lessons Learned from Open Source Projects for Facilitating Online Requirements Processes.” In , 240–255. Springer.

Li, Chuanyi, Liguo Huang, Jidong Ge, Bin Luo, and Vincent Ng. 2018. “Automatically Classifying User Requests in Crowdsourcing Requirements Engineering.” *Journal of Systems and Software* 138 (April): 108–123. doi:10.1016/j.jss.2017.12.028.

Liu, Lin, Qing Zhou, Jilei Liu, and Zhanqiang Cao. 2017. “Requirements Cybernetics: Elicitation Based on User Behavioral Data.” *Journal of Systems and Software* 124: 187–194.

Lützenberger, Johannes, Patrick Klein, Karl Hribernik, and Klaus-Dieter Thoben. 2016. “Improving Product-Service Systems by Exploiting Information From The Usage Phase. A Case Study.” *Procedia CIRP* 47: 376–381.

McKay, Alison, and Saikat Kundu. 2014. “A Representation Scheme for Digital Product Service System Definitions.” *Advanced Engineering Informatics* 28 (4): 479–498. doi:10.1016/j.aei.2014.07.004.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. “Efficient Estimation of Word Representations in Vector Space.” *ArXiv Preprint ArXiv:1301.3781*.

Misaghian, Negin, and Homayun Motameni. 2018. “An Approach for Requirements Prioritization Based on Tensor Decomposition.” *Requirements Engineering* 23 (2): 169–188.

Murray, Paul W, Bruno Agard, and Marco A Barajas. 2018. “Forecast of Individual Customer’s Demand from a Large and Noisy Dataset.” *Computers & Industrial Engineering* 118: 33–43.

Pacheco, Carla, Ivan García, and Miryam Reyes. 2018. “Requirements Elicitation Techniques: A Systematic Literature Review Based on the Maturity of the Techniques.” *IET Software* 12 (4): 365–378.

Papanagnou, Christos I, and Omeiza Matthews-Amune. 2018. “Coping with Demand Volatility in Retail Pharmacies with the Aid of Big Data Exploration.” *Computers & Operations Research* 98: 343–354.

Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. 2014. “Deepwalk: Online Learning of Social Representations.” In , 701–710. ACM.

Pohl, Klaus, and Ernst Sikora. 2007. “COSMOD-RE: Supporting the Co-Design of Requirements and Architectural Artifacts.” In , 258–261. IEEE.

Shimomura, Yoshiki, Yutaro Nemoto, Takatoshi Ishii, and Toshiyuki Nakamura. 2018. “A Method for Identifying Customer Orientations and Requirements for Product–Service Systems Design.” *International Journal of Production Research* 56 (7): 2585–2595.

Szwejczewski, Marek, Keith Goffin, and Zissis Anagnostopoulos. 2015. “Product Service Systems, after-Sales Service and New Product Development.” *International Journal of Production Research* 53 (17): 5334–5353.

Tukker, Arnold, and Ursula Tischner. 2017. *New Business for Old Europe: Product-Service Development, Competitiveness and Sustainability*. Routledge.

Valencia Cardona, AM, R Mugge, JPL Schoormans, and HNJ Schifferstein. 2014. “Challenges in the Design of Smart Product-Service Systems (PSSs): Experiences from Practitioners.” In *Proceedings of the 19th DMI: Academic Design Management Conference*. London, UK: Design Management Institute.

Wang, Yahui, Suihuai Yu, and Ting Xu. 2017. “A User Requirement Driven Framework for Collaborative Design Knowledge Management.” *Advanced Engineering Informatics* 33 (August): 16–28. doi:10.1016/j.aei.2017.04.002.

Wang, Zuoxu, Chun-Hsien Chen, Pai Zheng, Xinyu Li, and Li Pheng Khoo. 2019. “A Novel Data-Driven Graph-Based Requirement Elicitation Framework in the Smart Product-Service System Context.” *Advanced Engineering Informatics* 42: 100983.

Xu, Zheng, Lin Mei, Kim-Kwang Raymond Choo, Zhihan Lv, Chuanping Hu, Xiangfeng Luo, and Yunhuai Liu. 2018. “Mobile Crowd Sensing of Human-like Intelligence Using Social Sensors: A Survey.” *Neurocomputing* 279: 3–10.

Zheng, Maokuan, Xinguo Ming, Liya Wang, Dao Yin, and Xianyu Zhang. 2017. “Status Review and Future Perspectives on the Framework of Smart Product Service Ecosystem.” *Procedia CIRP* 64: 181–186. doi:10.1016/j.procir.2017.03.037.

Zheng, Pai, Chun-Hsien Chen, and Suiyue Shang. 2019. “Towards an Automatic Engineering Change Management in Smart Product-Service Systems–A DSM-Based Learning Approach.” *Advanced Engineering Informatics* 39: 203–213.

Zheng, Pai, Tzu-Jui Lin, Chun-Hsien Chen, and Xun Xu. 2018. “A Systematic Design Approach for Service Innovation of Smart Product-Service Systems.” *Journal of Cleaner Production*.

Zheng, Pai, Yang Liu, Fei Tao, Zuoxu Wang, and Chun-Hsien Chen. 2019. “Smart Product-Service Systems Solution Design via Hybrid Crowd Sensing Approach.” *IEEE Access* 7: 128463–128473.

Zheng, Pai, Zuoxu Wang, and Chun-Hsien Chen. 2019. “Industrial Smart Product-Service Systems Solution Design via Hybrid Concerns.” In *Procedia CIRP*. Zhuhai & Hongkong, China. doi:10.1016/j.procir.2019.02.129.

Zheng, Pai, Zuoxu Wang, Chun-Hsien Chen, and Li Pheng Khoo. 2019. “A Survey of Smart Product-Service Systems: Key Aspects, Challenges and Future Perspectives.” *Advanced Engineering Informatics* 42: 100973.