Towards an automatic engineering change management in smart product-service systems – A DSM-based learning approach

Zheng, Pai; Shang, Suiyue; Chen, Chun-Hsien

2019

Zheng, P., Chen, C.- H., & Shang, S. (2019). Towards an automatic engineering change management in smart product-service systems - A DSM-based learning approach. *Advanced Engineering Informatics*, 39203-213. doi:10.1016/j.aei.2019.01.002

<https://hdl.handle.net/10356/89051>

<https://doi.org/209549>;

<https://doi.org/10.1016/j.aei.2019.01.002>;

<https://doi.org/209549>

© 2019 Elsevier. All rights reserved. This paper was published in *Advanced Engineering Informatics* and is made available with permission of Elsevier.

Downloaded on 26 Apr 2023 07:48:18 SGT

Towards an Automatic Engineering Change Management in Smart Product-Service Systems – A DSM-based Learning Approach

Pai Zheng^{a, b1}, Chun-Hsien Chen^a, Suiyue Shang^a

^a*School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore 639798*

^b*Delta-NTU Corporate Laboratory for Cyber-Physical System, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798*

Abstract

The rapid development and implementation of smart, connected products (SCPs) in the engineering field has triggered a promising manufacturing paradigm of servitization, i.e. smart product-service systems (Smart PSS). As a complex solution bundle in both system and product level, its engineering change management differs from the existing ones mainly in two aspects. Firstly, massive in-context stakeholder-generated/product-sensed data during usage stage can be leveraged to enable its success in a data-driven manner. Secondly, the digitalized services, consisting of both hardware and software solutions, can also be changed in a more flexible way other than the physical components alone. Nevertheless, scarcely any work reports on how to conduct engineering change in such context, let alone a systematic approach to support the automatic generation of its change prediction or recommendation. Aiming to fill these gaps, this work proposes an occurrence-based design structure matrix (DSM) approach together with a three-way based cost-sensitive learning approach for automatic engineering change management in the Smart PSS environment. This informatics-based research, as an explorative study, overcomes the subjectivity and tedious assessment of the experts in the conventional approaches, and can offer useful guidelines to the manufacturing companies for managing their engineering changes for product-service innovation process.

Keywords: *product service systems; design structure matrix; engineering change management; data-driven design; three-way decision making; digitalization*

1. Introduction

Engineering change management is a critical issue in the complex system engineering field. Designers and manufacturers often face the situations of upgrading or changing product components/modules so as to meet individual customer's ever-changing requirements [1]. To address this issue, modular analysis or module partition, serves as the fundamental basis to enable complex engineering change process (e.g. aircraft assembly tooling change [2]). It is of great practical value to meet various customized needs, shorten the product development span and facilitate product re-design, upgrade and maintenance cost-effectively [3]. Among the module partition methods, design structure matrix (DSM), as a prevailing tool to manage engineering changes [4], helps to identify the interactions between groups of components, and cluster them into different modules with little inter-dependencies [1] ever since first coined in 1980s [5]. Hence, engineering changes can be undertaken within each module in an adaptable manner without affecting others. Despite its effectiveness, most existing DSM works

¹ Corresponding author: ericzhang@ntu.edu.sg

undertake direct coupling analysis based on experts/users' justification [6], which is subjective and time consuming, and there lacks an effective manner to automate the engineering change management process.

With the rapid development of information and communication technologies (ICT), a promising market of information densely product, i.e. smart, connected product (SCP) is ever-increasingly introduced by companies in the engineering field [7]. Except for the ubiquitous connectivity of Internet-of-Things (IoT) at low cost, SCP represents the third-wave of IT innovation, where value is created by embedding IT into the product itself [8]. Owing to its strong capabilities of collect, process, and even generate information, SCP serves as the medium and tool, to communicate with other products with intelligence, as well as to generate new services as add-on values [9]. Hence, an emerging paradigm, viz. Smart Product Service Systems (Smart PSS) was proposed, which represents today's manufacturing emphasis towards an IT-driven service-oriented manner, i.e. *servitization* [10]. Smart PSS, as a complex system including various product components (e.g. physical components, smart components, connectivity components) and service modules (e.g. software modules, hardware and software solutions), its engineering change management differs from the existing ones mainly in two aspects:

1) *In-context change management*. Conventional engineering change only considers marketing strategies (e.g. questionnaire, focus group, etc.) in the early design stage; or user feedback (e.g. online comments, review, etc.) in the usage stage to drive its design innovation (engineering change). They lack the real-time context awareness, and may not be sufficient to discover the latent needs. Conversely, in the smart, connected environment, both real-time sensing data and user feedback can be tracked and merged coherently in the context for better engineering change decision making.

2) *Digitalized service change*. Engineering change in the conventional view only changes the physical components/modules of the product. Nevertheless, in the Smart PSS context, the digitalized services, as a software and hardware bundle, are integrated with the physical components/modules to give better performance and user experience. Hence, engineering change should take the virtualized software modules into consideration as well.

Nevertheless, to the authors' knowledge, little work has been done to address this issue in the Smart PSS environment, let alone an effective approach to automate the process. To fill the above gaps, this work proposes an effective approach to support module partition process and to automate the engineering change management in the Smart PSS environment. The modular analysis and robustness evaluation process are conducted based on the co-occurrence of models in the design structure matrix (DSM), following the complex community partition method. Furthermore, its engineering change solution can be automatically predicted by a novel three-way based cost-sensitive learning approach. The rest of the paper is organized as follows: Section 2 provides a comprehensive review of related works. Section 3 introduces the composition of Smart PSS to depict its inherent changeable models. Then, Section 4 proposed the systematic approach including both a co-occurrence-based DSM support for modular analysis and robustness evaluation process, and a three-way based cost-sensitive learning approach for the automatic generation of change predictions. To validate the method, an illustrative example of a complex system maintenance service is given in Section 5. Finally, the conclusions, limitations and future works are highlighted in Section 6.

2. Related works

To give an overview of the background and further derive the research gaps, this section provides a comprehensive review of the related works on IT-driven PSS, and DSM-based engineering change management aspects.

2.1 IT-driven PSS

Goedkoop [11] first proposed PSS as “a system of products, services, networks of ‘players’ and supporting infrastructure that continuously strives to be competitive, satisfy customer needs and have a lower environmental impact than traditional business models”. Since then, many research works have been done to define its features, benefits, applications, design methodologies and system implementations [12]. In the PSS context, intangible services other than exclusively tangible products are integrated as a service bundle, to fulfill functionalities for customers, as well as to generate economic value for manufacturers and service providers. By adopting PSS business model with sustainability concerns, manufacturing companies have progressively shifted from the product-centered manner to the service-oriented one, i.e. servitization, where product is the tool generated for service innovation [13].

Table 1. Typical types of IT-driven PSS paradigms

Paradigm	Authors	Definition
Digitalized PSS	Lerch and Gotsch (2015) [14]	<i>“Manufacturers at this stage not only provide complex PSS to their customers, but also incorporate ICT solutions as a novel component in the product-service bundle, creating intelligent, independent operating systems that deliver the highest level of availability possible and optimize operations while reducing resource inputs.”</i>
Cyber-Physical PSS	Wiesner, Marilungo, and Thoben (2016) [15]	<i>“A combination of CPS technology with PSS business models.”</i>
Smart PSS	Valencia et al. (2015) [16]	<i>“The integration of smart products and e-services into single solutions delivered to the market to satisfy the needs of individual consumers.”</i>
	Kuhlenkötter et al. (2017) [17]	<i>“Smart PSS result from the digitalization of products and services as the digital connectivity between components allow their autonomous interaction and further development.”</i>
	Zheng et al. (2018) [10]	<i>“An IT-driven value co-creation business strategy consisting of various stakeholders as the players, intelligent systems as the infrastructure, smart, connected products (SCP) as the media and tools, and their generated services as the key values delivered that continuously strives to meet individual customer needs in a sustainable manner”</i>

Meanwhile, with the rapid development of ICT (e.g. IoT, Cyber-Physical Systems (CPS)), low cost, high performance embedded components are ever increasingly introduced by manufacturers, so as to embracing a promising market of information densely product, i.e. SCP [7]. Apart from the mechanism of IoT by providing a ubiquitous connectivity at a low cost for information transmitting, SCP represents the third-wave of IT competition. Owing to its unique capabilities, SCP as the medium and tool, enables the active interactions between

manufacturers and end-users, and also the generation of new services [18]. Hence, several new IT-driven servitization paradigms, i.e. Digitalized PSS, Cyber-Physical PSS, and Smart PSS have been proposed, which share similar visions, and their key definitions are summarized in Table 1.

McKay and Kundu [19] first depicted a computer-aided representation scheme to support PSS, viz. digital PSS which still lacks intelligence and connectedness. Then, digitalized PSS [20] underlined the trends of servitization and digitalization, which can be further classified into three types along its lifecycle, viz. smart service delivery (maintenance and repair stage), smart product optimization (production stage), and digital brain (R&D and design stage). Similarly, Cyber-Physical PSS [15] was proposed as an integration of CPS techniques and PSS business model to arise a lot of novel services and functions combining embedded systems with controlled behaviors, by connecting physical space with virtual one. Valencia et al. [16] gives a more general definition that Smart PSS is the integration of smart products and its generated e-services as a bundle. Moreover, as pointed out by [17], except for the IT-driven innovation, the digitalization of products and services should also be considered for Smart PSS new solution design as well. Hence, one can find that though lacking a unified terminology and its definition, all the concept shares some similar ideas, i.e. digitalization, servitization, and value co-creation enabled by the SCPs and IT [10]. This research follows the definition of our previous work and utilizes the term Smart PSS consistently throughout the paper.

2.3 DSM-based engineering change management

To manage the complex engineering changes, it is widely accepted that product modular analysis, or module partition is an effective mean [21]. It is of great practical value to meet various customized needs, shorten the product development span and facilitate product re-design, upgrade and maintenance cost-effectively [3]. Modular analysis identifies highly interactive groups of components/models to form modules, while maintaining little dependency between each two modules [4]. To achieve this, many approaches have been proposed, which can be further divided into two categories: graph theories and hierarchical clustering methods [22]. Among them, DSM has proved to be an efficient matrix-based approach ever since it coined [5] as a branch of graph theory [23]. It has been applied in many areas such as design change propagation, organization management, etc., and one can refer to [6] for more details. DSMs enable manufacturers to analyse both single and multiple domain architectures of engineering changes in a visualized manner [24]. To generate DSMs, it is necessary to understand the interactions between each model. Most of the existing studies utilized surveys, structured interviews, meetings with engineers, etc. based on the experts/users' opinions [25] which are subjective. Meanwhile, many of the reported works use binary or ordinal values to determine the level/type of dependency between models, which could result in inaccurate assessment due to the granularity and resolution [26]. Hence, it is necessary to discover alternative means to generate DSMs more objectively and accurately in a data-driven manner. Meanwhile, as pointed out by Gopsill et al. [25], elaborative input from users/experts are time consuming and not in-context, and an automatic prediction mechanism [27] should be provided to support new large complex products/systems change management.

To address the above issue, complex network theory [28] provides a potential way by looking at modular analysis in a transactional and continuous manner, as the community partition problem [29]. Hence, unlike the DSM, as a direct coupling method by analysing the relationships between models through matrix transformation, it should be adapted as an indirect coupling analysis method utilizing the probability to determine the interdependencies in a hierarchical clustering manner [3]. Meanwhile, digitalized services, as a software and

hardware solution bundle should also be included in the engineering change management process. Nevertheless, to the authors' knowledge, scarcely any work discusses about them in the Smart PSS context, let alone an automatic mean to manage engineering change predictions or recommendations.

3. Composition of Smart PSS

Except for a value co-creation proposition with active interactions among stakeholders [10], the composition of Smart PSS can be depicted from both the macro level, viz. system level, and the micro level, viz. product level respectively, as shown in Figure 1.

Figure 1. Composition of Smart PSS at system and product levels

System level. As an integrated eco-system, Smart PSS mainly consists of three key elements, viz. various SCPs as the media and tool, generated smart services as the add-on value, and their adaption in different smart environments, as shown in Figure 1(a) [30]. In this context, SCP represents the third wave of IT-driven competition, which value is created by embedding IT in the product itself [7]. With built-in sensors, microprocessors, connectivity components, etc., it occupies the capability of offline smartness, which can self-react to the context by exploiting the embedded systems. Meanwhile, by adopting SCPs in the smart environment, i.e. the one that is able to acquire and apply knowledge about an environment and adapt to its context-of-usage with better experience [31], the system-of-the-systems can be established. Hence, SCPs communicate with each other, adapted at a system level autonomously based on intelligent algorithms and big data [32], which represent the online smartness. Moreover, manufacturing companies are ever-increasingly servitizing their products with value-added services as a bundle [33], where the goal is to deliver other than a product (by ownership), but its performances and usage as an integrated solution [34]. Hence, companies can achieve more sustainability and an extended product lifecycle by rather upgrading/changing its intangible services than changing the physical components.

Product level. As a product-service bundle, Smart PSS is composed by three parts, i.e. physical components, smart and connectivity components, and SCP generated services [10], as shown in Figure 1(b). The first two are the inherent elements of a SCP [7] where 1) *physical components* comprising the electrical and mechanical parts as usual product's; 2) *smart components* containing the sensors, microprocessors, data storage, controls, software,

and, typically, an embedded operating system and enhanced user interface. In many cases, software allows a single physical device or replaces some hardware components to perform at multi-levels [18]; and 3) *connectivity components* which comprise the ports, antennae, and protocols enabling wired or wireless connections with the product. Meanwhile, the SCP generated services are the add-on values which mainly include three folders: 1) usual product-service bundle without intelligence and connectedness (e.g. offline maintenance service); 2) smart e-services, which are independent with the product components (e.g. broadcasting APPs in the smartphone); and 3) smart digitalized services, which are the ones largely dependent with the product itself (e.g. digital twin-enabled machine tool).

From the above description, it can be found that engineering change management in the Smart PSS context refers to the digitalized product-service bundle rather than the e-services generated with little concern to the product itself. Meanwhile, it is conducted in the usage stage where both product and its services are created, rather than in the SCP design stage from the very beginning of lifecycle. Therefore, engineering change management is enabled by the state-of-the-art digitalization [35] and artificial intelligence techniques, where massive stakeholder-generated (both consumers' and manufactures') and SCP-sensed data serve as the key to determine the final change decisions. At the same time, owing to the inter-connectedness established between the components and its virtual twin, changes can be easily monitored, analyzed and even predicted in a data-driven manner. These differs from the existing scenarios and serve as the fundamentals of the proposed approach in the following section.

4. Proposed data-driven engineering change management process

Based on the description above, the flowchart of each main step and its corresponding diagram of the proposed data-driven engineering change management approach is depicted in Figure 2. The core idea is to cluster modules based on the existing co-occurrence information as the input, and to recommend the most appropriate engineering change modules based on the proposed three-way cost-sensitive learning approach. Four key steps can be categorized as: 1) data collection and pre-processing; 2) establishment of co-occurrence-based DSM; 3) modular analysis and its robustness evaluation; 4) three-way-based learning, which are further introduced below.

4.1 Data collection and pre-processing

In the Smart PSS context, various hardware sensed data, viz. built-in sensors in SCPs (e.g. barometer) and massive stakeholder-generated design data (e.g. ratings, comments) from heterogenous data sources (i.e. both online and offline) can be leveraged as the condition dataset. Correspondingly, manufacturers/service providers' service logs (e.g. component change records) as the decision dataset can be utilized to keep the historical engineering change records. Hence, a proper information table needs to be established to describe the existing change management in a pre-defined format. In this work, a 4-tuple information table is generated as:

$$T = (R, A, V, f) \quad (1)$$

, where $R = \{r_1, r_2, \dots, r_{|U|}\}$ is a nonempty set of total historical records, as the universe. $A = \{a_1, a_2, \dots, a_{|A|}\}$ is a nonempty finite set of attributes and $\forall a \in \{C_H, C_S, D_S\}$, where C_H is the set of hardware sensor attributes (e.g. location, temperature), and C_S is the set of social sensor attribute (e.g. user comments), as shown in Table 2. They together form the *condition set*. Meanwhile, D_S is the set of engineering change decisions made, denoted as the *decision set*, where $D_S = \{m_1, m_2, \dots, m_{|K|}\}$, m stands for the model occurred in this record and K is the total number of models, respectively, as shown in Table 1. $A = C_H \cup C_S \cup D_S$, and $C_H \cap C_S \cap D_S = \emptyset$. V_a is a nonempty set of

values for an attribute $a \in A$. $f: R \times A \rightarrow V$ is an information function, where $f(x_i, a_j) = v_{ij}$ ($i = 1, 2, \dots, |R|, j = 1, 2, \dots, |A|$) denotes the attribute value of object r_i under a_j .

Moreover, in the information table, the values of attributes can be heterogeneous, including numerical data (e.g. temperature, ratings) or non-numerical data (e.g. text, video and audio). For the prior one, specific ranges should be pre-defined to deal with the high variety of discrete number by categorizing them into different classes [36]. Meanwhile, for the latter one, specific semantics should be extracted and again categorized based on the information fusion techniques, and one may refer to [37] for more details.

Figure 2. Flowchart and its diagrams of proposed data-driven engineering change management process

4.2 Establishment of co-occurrence-based DSM

Based on the information table, the co-occurrence of models can be generated in the DSM, as a symmetric matrix shown in Table 3. Unlike existing DSM partitioning techniques performed either by the binary or ordinal data based on the experts' subjective opinions, in contrast, it is depicted by the times of co-occurrence between each two models of the complex system. For example, the frequency of co-occurrence between m_1 and m_2 is 0. Hence, co-occurrence-based DSM measurements are transactional and continuous [25], as well as more objective in nature by computing within the historical dataset.

Table 2. A simplified example of engineering change information table

Record No.	Condition set			Decision set (D_S)
	C_{H1}	C_{H2}	C_{S1}	$Models$
1	0	0	1	m_1, m_2
2	1	0	0	m_3
3	0	1	0	m_1, m_8
4	1	0	0	m_1, m_4, m_5, m_8
5	0	1	0	m_1
6	0	0	1	m_1, m_6, m_8
7	0	1	0	m_1, m_8
8	1	1	0	m_2, m_4, m_8

Table 3. An example of co-occurrence-based DSM

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
m_1		0	0	1	1	1	1	4
m_2	0		0	1	0	0	0	1
m_3	0	0		0	0	0	0	0
m_4	1	1	0		1	0	0	2
m_5	1	0	0	1		0	0	1
m_6	1	0	0	0	0		0	1
m_7	1	0	0	0	0	0		0
m_8	4	1	0	2	1	1	0	

4.3 Modular analysis and its robustness evaluation

In the co-occurrence-based DSM, the modularization of components is very similar to the case of community partitions in the complex social network [38], where each model (i.e. component) represents the node, and each occurrence denotes the edge with its frequency as the weight. Hence, the Louvain community-partitioning algorithm [39], as a weighted undirected graph-based approach is adopted in this research, where the final goal is to generate a set of partitions for the DSM with the highest modularity value. According to Newman (2006), the modularity (Q), as a scalar value between 1 and -1, represents the edge (viz. occurrence) density inside communities as compared to the edge density between the community. The larger Q represents the better quality of the partition, and it is defined as follows:

$$Q = \frac{1}{2N} \sum_{i,j} \left[W_{ij} - \frac{S_i S_j}{2N} \right] \delta(C_i, C_j) \quad (2)$$

, where W_{ij} denotes the weighted co-occurrence between model i and j . S_i denotes the number of models that have co-occurrences with model vertex i and it can be written as $S_i = \sum_j W_{ij}$. N represents the number of co-occurrences within the matrix, $N = \sum_{i,j} W_{ij}$. C_i represents the complex system that include i , and δ is the Kronecker delta function where:

$$\delta(a, b) = \begin{cases} 1, & \text{when } a = b \\ 0, & \text{when } a \neq b \end{cases} \quad (3)$$

To obtain the highest modularity partition, Louvain algorithm iterates between two steps, which is depicted

in **Algorithm 1**. Step 1 is to assign each model as a separate partition, and followed by sequentially moving one model to a different partition with the calculation of the change value of modularity (ΔQ), denoted as:

$$\Delta Q = \left[\frac{\sum I + S_{i,in}}{2N} - \left(\frac{\sum T + S_i}{2N} \right)^2 \right] - \left[\frac{\sum T}{2N} - \left(\frac{\sum T}{2N} \right)^2 - \left(\frac{S_i}{2N} \right)^2 \right] \quad (4)$$

, where $S_{i,in}$ is the sum of the weights of co-occurrence from model i to model in in the matrix. Hence, the maximum modularity change can be identified. Then, Step 2 groups the models in the same partitioned module, and combines the co-occurrences of model activity to form single co-occurrence links to the rest of the matrix. And the internal co-occurrences between models of the same module are represented by self-loops. Hence, it can be seen as a form of hierarchical clustering, where the second step iteration ends until the modularity reached the highest value, that is no partition will be conducted. As the example shown in Table 4, it was partitioned as $M = 4$, viz. $\{(1, 6, 7, 8), (2, 4, 5), (3)\}$ with a modularity value of $Q = 0.079$.

Algorithm 1. Louvain community partition method.

Input: Initial network G
Output: Communities (C), Modularity (Q), Final network (G')
Methods: Louvain Method

```

1   Load initial network  $G$ 
2   Put each node of  $G$  in its own community with index  $C$ 
3   Set initial modularity  $q = -1$ 
4   While  $q < Q(G, C)$  do
5        $q = Q(G, C)$ 
6       //Step 1 of Louvain method: Move nodes
7       While one or more nodes are moved do
8           For random node  $n$  of  $G$  do
9                $best\_q = -1$ 
10               $best\_c = \text{community of } n$ 
11              For all neighbor nodes  $v$  of  $n$  do
12                   $add\_q = \Delta Q$ 
13                  If  $best\_q < add\_q$  then
14                       $best\_q = add\_q$ 
15                       $best\_c < \text{community of } n$ 
16                  End if
17              End for
18               $C = \text{Place } n \text{ in the } best\_q$ 
19          End for
20      End while
21      Return  $C$ 
22      // Step 2 of Louvain method: Aggregation
23       $G' = \text{aggregate nodes in the same community based on } C$ 
24      Return  $G'$ 
25  End while
26  Return  $Q = best\_q$ 

```

To guarantee the robustness of the module partition value (Mv) in the co-occurrence-based matrix, two criteria should be met with the existing matrix: 1) adding/deleting any one of the engineering change record will not affect the value Mv , and 2) the change of its modularity should be less than a threshold value, ξ where:

$$\xi = \max\left(\frac{|Q - Qt|}{Q}\right) \quad (5)$$

Q stands for the calculated modularity of existing records, while Qt stands for the one after adding or deleting any one record, and it is recommended that $\xi < 0.05$. For example, the record No.4 is deleted from Table 2, and

the calculated $\xi = 1.27$, and the $Mv = 4$, viz. $\{(1, 6, 7, 8), (2, 4, 5), (3)\}$, which is obviously not a robust partition for the engineering change management.

Table 4. An example of modularized DSM

	m_1	m_6	m_7	m_8	m_2	m_4	m_5	m_3
m_1		1	1	4	0	1	1	0
m_6	1		0	1	0	0	0	0
m_7	1	0		0	0	0	0	0
m_8	4	1	0		1	2	1	0
m_2	0	0	0	1		1	0	0
m_4	1	0	0	2	1		1	0
m_5	1	0	0	1	0	1		0
m_3	0	0	0	0	0	0	0	

4.4 Three-way based learning approach for engineering change recommendation

4.4.1 Module-based engineering change information table

Based on the above modular analysis with robustness, engineering change models (m) in each historical record can be clustered into different modules (M), as the example shown in Table 5. Furthermore, these categorized records can be further utilized to recommend or predict design modular changes based on the following proposed three-way based learning approach with established cost matrix.

Table 5. A simplified example of engineering change information table

Record No.	Condition set			Decision set (D_S)
	C_{H1}	C_{H2}	C_{S1}	M
1	0	0	1	M_1, M_2
2	1	0	0	M_3
3	0	1	0	M_1
4	1	0	0	M_1, M_2
5	0	1	0	M_1
6	0	0	1	M_1
7	0	1	0	M_1
8	1	1	0	M_1, M_2

4.4.2 Three-way decision theoretical rough set model

After each record has been transformed into a module-based manner consistently, according to the rough set theory [40], the indiscernibility relation of the subset of condition attributes $A_S \in A$ is defined as:

$$\text{IND}(A_S) = \{(x, y) \in U \times U \mid \forall a \in A_S, f(x, a) = f(y, a)\} \quad (6)$$

, where two objects x and y are indiscernible with respect to A_S if and only if they have the same value on every condition attribute in A_S . The partitioning of A_S induced by V is represented as:

$$A_S/V = \{c_1, c_2, \dots, c_N\} \quad (7)$$

, where $\forall c \in A_S/V$ is an equivalence class, and $\forall (c_i, c_j) \in A_S/V, c_i \cap c_j = \emptyset$. Furthermore, $P(M_k^i)$ represents the probability that module k to be changed based on its occurrence frequency in the i^{th} equivalent class:

$$P(M_k^i) = |D(M_k) \cap [c_i]| / |[c_i]| \quad (8)$$

, where $D(M_k) \subseteq U$ stands for all the objects that contain M_k in the decision attributes, $|\cdot|$ denotes the cardinality of a set. For example, record 1 and 6 in Table 5 are indiscernible with the same conditional attributes $c_l = (0, 0, 1)$ but different decision attributes, viz. $\{(M_1, M_2), (M_2)\}$, hence, the occurrence probabilities of M_1 and M_2 in the equivalence class c_l are calculated as $P(M_1^1) = 1$ and $P(M_2^1) = 0.5$, respectively.

Meanwhile, the result of conditional probability divides the U into three regions named as *occurred region* (OR), *boundary region* (BR) and *absent region* (AR), respectively. Given a pair of thresholds α and β , where $0 < \beta \leq \alpha < 1$, the OR, BR and AR regarding module k in class c_i are defined as follows:

$$OR(D(M_k)) = \{x \in U \mid P(D(M_k) \mid [c_i]) > \alpha\} \quad (9)$$

$$BR(D(M_k)) = \{x \in U \mid \beta \leq P(D(M_k) \mid [c_i]) \leq \alpha\} \quad (10)$$

$$AR(D(M_k)) = \{x \in U \mid P(D(M_k) \mid [c_i]) < \beta\} \quad (11)$$

The β and α are the lower and upper threshold of the boundary region, where probabilities below β are in the absent region, and the ones above α are in the occurred region, respectively. For example, in Table 5, if $1 > \alpha > \beta > 0.5$, then M_1 of c_l is in the occurred region while M_2 in the absent region.

4.4.3 Construction of random forest to predict module occurrence probability

Based on Eq. (8), $P(M_k^i)$ of existing dataset can be easily determined. Nevertheless, for those condition sets (e.g. (1, 1, 1)) not included in the existing information table, a random forest should be established based on the large volume training set by recursively partitioning the data, as depicted in **Algorithm 2**. These trees are merged together to form a random forest with module occurrence probabilities of each condition attribute set.

Algorithm 2. Pseudo code of constructing random decision tree

Input: Training dataset (D_T), Condition attributes (C), Probability (P)

Output: A Random Decision Tree (CNode)

Method: New Random Tree

```

1   If ( $C = \emptyset$  or  $D_T = \emptyset$ ) then // Check if the input dataset is null
2       return Null
3   End if
4    $P_M = P(D_T)$ ; // Calculate the probability of module occurrence ( $P_M$ )
5   CNode = NewRandomTree ( $D_T, C, P$ ); // Construct current node based on input  $D_T, C, P$ 
6   found = false; // Randomly select condition attributes and split (Line 6 – Line 13)
7   For ( $a \in C$ )
8        $C = C - \{a\}$ ;
9       If (InformationGain ( $a$ ) > 0) then
10           found = true;
11           break;
12       End if
13   End for
14   If (not found) then
15       CNode.child = Null;
16       Return CNode;
17   End if
18   CNode.splittingAttribute =  $a$ ; // Construct random tree (Line 18 – Line 26)
19    $N$  = number of attribute values of  $a$ ;
20   CNode.child = new
    bulidRandomTree [ $N$ ]
21   For ( $i = 1$  to  $N$ )
22        $D_T(i) = \{a \in D_T \mid a(x) = i\}$ 
23        $P_M = P(D_T(i))$ ;
24       CNode.children [ $i$ ] = bulidRandomTree ( $P_{D(i)}, C, P_M$ );
25   End for
26   Return CNode

```

Based on **Algorithm 2**, each random decision-tree produces a prediction result, i.e. $CNode.P$, based on the conditional/decision attributes and its values in the training set. By leveraging these probabilities, each module occurrence probability in the testing set $Pt(M_k^i)$ can be predicted, as shown in **Algorithm 3**.

Algorithm 3. Pseudo code of prediction based on random decision tree

Input: Current node (CNode), Test dataset (D_t)
Output: M Prediction Result (P_t)
Method: PredictionbyRandomDecisionTree (PRDT)

```

1    $a = CNode.splittingAttribute;$ 
2    $j = a(D_t);$ 
3   If (CNode.child = NULL) then
4       Return CNode.P;
5   Else if (CNode.child [ $j$ ] = NULL) then
6       Return CNode.P;
7   Else
8       Return PRDT (CNode.child [ $j$ ],  $D_t$ );
9   End if

```

4.4.3 Cost-sensitive learning

Once the probability of testing dataset is predicted, the next step is to determine the proper thresholds, i.e. α and β to categorize each predicted result into different region, and then compare it with the original decision attribute set of each testing record to reveal its prediction accuracy. The cost-sensitive learning approach is adopted in this work, where the cost matrix is shown in Table 6. The first column depicts the predicted result of classification in OR , BR , or AR , and the first row shows the actual classification of testing result, i.e. occurred (O) or not occurred $\sim O$ and $\lambda_i = \{\lambda_{oo}, \lambda_{oa}, \lambda_{bo}, \lambda_{ba}, \lambda_{ao}, \lambda_{aa}\}$ stands for the cost occurred. Generally, the misclassification of testing result will result in a punishment cost, i.e. λ_{oa} and λ_{ao} , and the classification of boundary region will also lead to a teaching cost, viz. λ_{bo} and λ_{ba} , where $\lambda_{bo} = \lambda_{ba}$. Meanwhile, the accurate classification will result in no cost, viz. $\lambda_{oo} = \lambda_{aa} = 0$.

Table 6. Cost matrix for prediction result

Action	O	$\sim O$
OR	λ_{oo}	λ_{oa}
BR	λ_{bo}	λ_{ba}
AR	λ_{ao}	λ_{aa}

According to [41], the threshold values of β and α can be calculated as:

$$\alpha = \frac{\lambda_{oa} - \lambda_{ba}}{(\lambda_{oa} - \lambda_{ba}) + (\lambda_{bo} - \lambda_{oo})} = \frac{\lambda_{oa} - \lambda_{ba}}{\lambda_{oa}} \quad (12)$$

$$\beta = \frac{\lambda_{ba} - \lambda_{aa}}{(\lambda_{ba} - \lambda_{aa}) + (\lambda_{ao} - \lambda_{bo})} = \frac{\lambda_{ba}}{\lambda_{ao}} \quad (13)$$

, where the equations can be further denoted as:

$$\lambda_{oa} = \frac{1}{1-\alpha} \lambda_{ba} \quad (14)$$

$$\lambda_{ao} = \frac{1}{\beta} \lambda_{ba} \quad (15)$$

From the above equations, one can obtain the total cost as follows:

$$T_c = \sum_{k=1}^M (\lambda_{oa} R_o^k + \lambda_{ba} R_b^k + \lambda_{ao} R_a^k) \quad (16)$$

, where R_o^k, R_b^k, R_a^k are the total numbers of predicted results where module k classified in the OR , BR and AP ,

respectively. The pre-defined threshold α and β are leveraged to compare with the probability value Pt to classify different categories based on the three-way model, viz. Eqs. (9-11). Then, the average cost A_{cost} can be calculated based on the cost matrix in Table 5 and Eq. (17), as shown in **Algorithm 4**.

$$A_{cost} = T_c / N \quad (17)$$

, where N is the total number of classes. Based on the calculation result, the combination of α and β with the lowest average cost should be selected by the manufacturer to ensure the accuracy of prediction. To this end, automatic engineering change management can be undertaken in a data-driven manner cost-effectively.

Algorithm 4. Pseudo code of average cost computation

Input: Condition attribute of the testing dataset (C_{At}), Thresholds (α, β), Probability of module occurrence (Pt)

Output: Average Cost (A_{COST})

Method: AverageCostCalculation

```

1  For ( $x \in C_{At}$ ) do
2    If (E is training dataset) then
3       $P(x)$  is calculated based on Eq. (8)
4    Else  $P(x)$  is predicted by the Algorithm 2
5    End if
6    O ( $x$ ) is the original modules of  $x$ 
7    For (M from 0 to N) do // Predict probability of each module (0 to N) separately
8      If ( $P(x) > \alpha$ ) then
9        If ( $O(x) = M$ ) then
10          $N_{AA}++$ ; // Number of accept-accept region
11        Else
12          $N_{AR}++$ ; // Number of accept-reject region
13        End if
14      Else if ( $P(x) > \beta$ ) then
15        If ( $O(x) = M$ ) then
16          $N_{RA}++$ ; // Number of reject-accept region
17        Else
18          $N_{RR}++$ ; // Number of reject-reject region
19        End if
20      Else
21        If ( $O(x) = M$ ) then
22          $N_{PA}++$ ; // Number of pending-accept region
23        Else
24          $N_{PR}++$ ; // Number of pending-reject region
25        End if
26      End if
27    End for
28  End for
29   $A_{COST}$  is calculated by Eq. (17)
30  return  $A_{COST}$ 

```

5. An illustrative example

In order to validate the feasibility of the proposed approach in a generic manner, as well as to maintain the confidentiality of the company information, a sum of 5250 maintenance service records of a complex smart water dispenser (SWD) are collected from company X (one can refer to supplementary materials). They are further simplified and pre-processed based on the pre-defined information table (*Step 1*), as shown in Table 7. The condition set consists of 3 real-time attributes detected from the built-in sensors, viz. C_1 : temperature; C_2 : flow rate; C_3 : pressure; and expert's assessment C_4 : failure mode. Among them, C_1 , C_2 , and C_3 only have three modes, i.e. -1 represents below normal range (e.g. temperature below 20 °C), 0 for normal range (e.g. flow rate is between 1.39-1.89Lpm), and 1 for above normal range (e.g. pressure is above 827kPa). Meanwhile, C_4 has four different

failure mode, i.e. general electrical problem (0); heater Malfunction (1); general mechanical problem (2); cooling malfunction (3). Each combination of them results in a change of models (components) in SWD (viz. decision set), which is documented in the maintenance service records, as shown in Table 7. SWD, as a Smart PSS, not only its physical component can be changeable (i.e. m_1 : Sediment trap), but also its software modules can be changed as a digitalized service (e.g. m_{20} : a control panel function upgrade). Due to its model complexity with different combination of condition/solution options, the company needs a cost-effective way to manage its existing records by modularizing its models, and further automate its engineering change prediction/recommendation.

Table 7. A small portion of A's service maintenance engineering change information table

Record No.	Condition set				Decision set
	C_1	C_2	C_3	C_4	Models
1	1	1	0	0	1, 2, 4, 6
2	1	1	1	1	3, 5, 7, 8, 9, 11, 20, 22
3	1	0	0	2	3, 5, 7, 8, 9, 11, 12, 15, 16, 17, 19
4	1	0	0	0	3, 5, 7, 8, 9, 11, 12, 15, 16, 17, 19
5	1	-1	0	1	3, 5, 7, 8, 9, 10, 13, 14, 18, 21
6	0	0	1	2	10, 13, 14, 18, 21
7	1	0	1	1	11, 12, 15, 16, 17, 19, 20, 22
8	1	1	1	1	3, 5, 7, 8, 9, 11, 20, 22
...	...	...	...	...	...

Step 2: Table 8 presents the co-occurrence-based DSM of A including 22 models, viz. $M_{i,j}$, where $1 \leq i, j \leq 22$. The index of each $M_{i,j}$ represents the co-occurrence frequency of i^{th} model and j^{th} model in the maintenance records, where 1705 is the maximum number of co-occurrence ($M_{2,4} M_{2,6}$), and 109 is the minimum one ($M_{1,15} M_{2,15} M_{4,15} M_{6,15}$). By utilizing the open graph viz platform, Gephi® (<https://gephi.org/>), one can derive the undirected graph of the 22 models without weights, as shown in Figure 3.

Figure 3. Co-occurrence-based model interaction in an undirected graph with no weights.

Step 3: For modularity analysis, this work utilized a Python® package, i.e. *NetworkX* with the community

application programming interface implementation of the Louvain community-partitioning algorithm [42] running in the Jupyter notebook web application written by Python 3 programming language. Hence, the modules and modularity values can be calculated where $M = 5$, viz. $\{[1, 2, 4, 6], [3, 5, 7, 8, 9], [10, 13, 14, 18, 21], [11, 12, 15, 16, 17, 19], [20, 22]\}$, and $Q = 0.33865$. To test the robustness of the module partition, any random record is deleted and added, and its calculated that $\xi = 0.012 < 0.05$ and $M = 5$, which proves the robustness of the module partition result. To show the level of interdependence of each two models in a more visualized manner, a weighted graph is further presented in Gephi® environment by dividing each co-occurrence number by the maximum one, as shown in Figure 4. The thickness of each edge denotes its interdependency or co-occurrence frequency between two models and the ones with strong connections are depicted with different colors, which matches well with the Louvain partition result.

Figure 4. Co-occurrence-based model interaction in a weighted undirected graph.

Step 4: After Step 3, the modularized information table can be derived, as shown in Table 9. The models in each service record are replaced by the transformed modules, i.e. $M = \{1, 2, 3, 4, 5\}$. Then, according to Algorithm 2, the original dataset is repeated 20 times with random partitioning (i.e. 20×cross-validation) to separate them into the training set (80%) as chosen, and testing set (20%) as the remaining to form the random decision tree. These trees are merged together to form a random forest with module occurrence probabilities of each condition attribute set. The *scikit-learn* python library is leverage to construct the random forest with the decision probability of each module in a specific condition set based on Algorithm 3. Hence, the predicted results of the testing dataset can be derived as partially shown in Table 10, where Pt_i corresponds to the occurrence probability of the i^{th} module in an equivalence class.

Pt of each class can be calculated to be compared with the α and β . The ones bigger than α are put into the occurred region, in-between α and β in the boundary region, and smaller than β in the absence region, respectively, based on Eqs. (9-11). According to the cost matrix in Table 6, this study defines the cost of $\lambda_{bo} = \lambda_{ba}$

= 10. Meanwhile, based on Eq. (14) and Eq. (15), the cost of λ_{oa} and λ_{ao} can be further determined based on the different combination of α and β , which are selected by an interval of 0.05 in this work (excluding situations where $\beta = 0$, or $\alpha = 1$).

Table 8. Co-occurrence-based DSM of the complex system A

Model	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1		1698	235	1698	235	1698	235	235	235	320	338	113	320	320	109	113	113	320	113	138	320	138
2	1698		235	1705	235	1705	235	235	235	320	338	113	320	320	109	113	113	320	113	138	320	138
3	235	235		2135	1463	235	1463	1456	1418	132	1264	319	132	132	331	331	331	132	331	156	132	156
4	1698	1705	235		235	1705	235	235	235	320	338	113	320	320	109	113	113	320	113	138	320	138
5	235	235	1463	235		235	1476	1469	1431	132	1277	332	132	132	344	344	344	132	344	156	132	156
6	1698	1705	235	1705	235		235	235	235	320	338	113	320	320	109	113	113	320	113	138	320	138
7	235	235	1463	235	1476	235		1469	1431	132	1277	332	132	132	344	344	344	132	344	156	132	156
8	235	235	1456	235	1469	235	1469		1424	132	1270	325	132	132	337	337	337	132	337	156	132	156
9	235	235	1418	235	1431	235	1431	1424		132	1232	311	132	132	323	323	323	132	323	132	132	132
10	320	320	132	320	132	320	132	132	132		130	116	1341	1341	116	116	116	1341	116	124	1341	124
11	338	338	1264	338	1277	338	1277	1270	1232	130		1272	130	130	1260	1251	1284	130	1284	327	134	327
12	113	113	319	113	332	113	332	325	311	116	1272		116	116	1269	1260	1293	116	1293	171	120	171
13	320	320	132	320	132	320	132	132	132	1341	130	116		1341	116	116	116	1341	116	124	1341	124
14	320	320	132	320	132	320	132	132	132	1341	130	116	1341		116	116	116	1341	116	124	1341	124
15	109	109	331	109	344	109	344	337	323	116	1260	1269	116	116		1248	1281	116	1281	171	120	171
16	113	113	331	113	344	113	344	337	323	116	1251	1260	116	116	1248		1272	116	1272	171	120	171
17	113	113	331	113	344	113	344	337	323	116	1284	1293	116	116	1281	1272		116	1305	171	120	171
18	320	320	132	320	132	320	132	132	132	1341	130	116	1341	1341	116	116	116		116	124	1341	124
19	113	113	331	113	344	113	344	337	323	116	1284	1293	116	116	1281	1272	1305	116		171	120	171
20	138	138	156	138	156	138	156	156	132	124	327	171	124	124	171	171	171	124	171		128	1272
21	320	320	132	320	132	320	132	132	132	1341	134	120	1341	1341	120	120	120	1341	120	128		128
22	138	138	156	138	156	138	156	156	132	124	327	171	124	124	171	171	171	124	171	1272	128	

Table 9. Modularized information table of A

Record No.	Condition attributes	Decision attributes (Models)	Modules
1	[1, 1, 0, 0]	[1, 2, 4, 6]	[1]
2	[1, 1, 1, 1]	[3, 5, 7, 8, 9, 11, 20, 22]	[2, 4, 5]
3	[1, 0, 0, 2]	[3, 5, 7, 8, 9, 11, 12, 15, 16, 17, 19]	[2, 4]
4	[1, 0, 0, 0]	[3, 5, 7, 8, 9, 11, 12, 15, 16, 17, 19]	[2, 4]
5	[1, -1, 0, 1]	[3, 5, 7, 8, 9, 10, 13, 14, 18, 21]	[2, 3]
6	[0, 0, 1, 2]	[10, 13, 14, 18, 21]	[3]
7	[1, 0, 1, 1]	[11, 12, 15, 16, 17, 19, 20, 22]	[4, 5]
8	[1, 1, 1, 1]	[3, 5, 7, 8, 9, 11, 20, 22]	[2, 4, 5]
...	...	...	...

Table 10. Information table of testing dataset and its predict result of each module

Record No.	Condition Set				Decision Set					Probability of M				
	C_1	C_2	C_3	C_4	M_1	M_2	M_3	M_4	M_5	Pt_1	Pt_2	Pt_3	Pt_4	Pt_5
1	1.00	1.00	0.00	2.00	1.00	0.00	0.00	0.00	0.00	0.81	0.21	0.16	0.02	0.02
2	1.00	0.00	1.00	3.00	1.00	0.00	0.00	0.00	1.00	0.68	0.44	0.00	0.76	0.76
3	0.00	1.00	0.00	3.00	1.00	0.00	1.00	0.00	0.00	0.38	0.00	0.29	0.00	0.00
4	1.00	1.00	1.00	2.00	0.00	0.00	1.00	0.00	1.00	0.00	0.87	0.13	0.86	0.86
5	0.00	1.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.34	0.00	0.25	0.01	0.01
6	0.00	1.00	0.00	1.00	1.00	0.00	0.00	0.00	0.00	0.37	0.00	0.29	0.00	0.00
7	0.00	0.00	1.00	3.00	1.00	0.00	1.00	0.00	0.00	0.42	0.00	0.78	0.00	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Hence, according to Algorithm 4 and Eq. (17), the average cost can be calculated, as shown in Table 11. Meanwhile, the scatter plot (Figure 5(a)) and surface plot (Figure (b)) of the average cost (coordinate in z-axis) by each combination of α (coordinate in x-axis) and β (coordinate in y-axis) are also drawn in the MATLAB 2017a environment to show its tendency more clearly. It can be found that average cost is somewhat decreasing with higher value of α and β . The lowest average cost is 19.21 when $\beta = \alpha = 0.9$, which reveals the most accurate classification result in the three-way cost-sensitive learning model. Therefore, this combination together with the prediction result can be adopted in the automatic generation of engineering change solutions in the A.

Figure 5. Plot of the average cost of testing dataset based on different combination of α and β .

Table 11. Average cost of each combination of α and β in the testing set of A

α / β	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
0.05	34.28	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.10	35.36	28.05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.15	36.53	29.21	26.44	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.20	37.55	30.23	27.46	24.92	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.25	38.80	31.49	28.71	26.17	24.92	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.30	39.80	32.49	29.72	27.18	25.93	24.14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.35	40.73	33.42	30.64	28.11	26.86	25.07	23.35	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.40	41.66	34.35	31.58	29.04	27.79	26.00	24.28	22.75	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.45	42.80	35.49	32.71	30.17	28.93	27.14	25.42	23.89	22.76	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.50	43.96	36.65	33.87	31.33	30.09	28.30	26.58	25.05	23.92	22.80	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.55	45.35	38.03	35.26	32.72	31.47	29.69	27.96	26.44	25.30	24.19	23.25	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.60	47.03	39.72	36.94	34.41	33.16	31.37	29.65	28.12	26.99	25.87	24.93	24.12	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.65	48.43	41.12	38.34	35.80	34.56	32.77	31.05	29.52	28.39	27.27	26.33	25.52	24.41	NaN	NaN	NaN	NaN	NaN	NaN
0.70	50.11	42.80	40.03	37.49	36.24	34.45	32.73	31.21	30.07	28.95	28.01	27.20	26.10	25.08	NaN	NaN	NaN	NaN	NaN
0.75	51.51	44.20	41.43	38.89	37.64	35.85	34.13	32.61	31.47	30.35	29.41	28.60	27.50	26.48	25.27	NaN	NaN	NaN	NaN
0.80	52.94	45.63	42.85	40.31	39.07	37.28	35.56	34.03	32.90	31.78	30.84	30.03	28.93	27.90	26.70	25.48	NaN	NaN	NaN
0.85	52.33	45.02	42.24	39.71	38.46	36.67	34.95	33.42	32.29	31.17	30.23	29.42	28.32	27.29	26.09	24.87	23.26	NaN	NaN
0.90	49.85	42.53	39.76	37.22	35.97	34.18	32.46	30.94	29.80	28.69	27.75	26.93	25.83	24.81	23.60	22.38	20.78	19.21	NaN
0.95	50.84	43.52	40.75	38.21	36.96	35.18	33.45	31.93	30.79	29.68	28.74	27.92	26.82	25.80	24.59	23.37	21.77	20.20	19.24

6. Conclusion

Engineering change management plays a significant role in leveraging design efficiency, shorten system development time span, and reducing resource waste. Most existing studies undertake a direct or indirectly module partition approaches by clustering various components/models based on the experts/users' domain knowledge, which is subjective and time consuming. In the smart, connected environment, massive stakeholder-generate/product-sensed data can produce large amount of in-context information, and hence should be leveraged to manage changes of the complex digitalized product/service in a data-driven manner. Nevertheless, few works discuss about this issue, let alone an automatic approach to predict/recommend changes under certain conditions. To fill the gaps, this work proposed a novel approach to support engineering change management in the Smart PSS context. Its main contributions can be concluded as:

1) Established a formatted information table with systematic flowchart to automate engineering change management in a generic way. Historical datasets are classified into condition set (e.g. sensing or user-generated datasets) and decision set (e.g. service providers' comments) with inherent equivalence classes defined. Meanwhile, by following the proposed flowchart, scenarios such as service maintenance, component upgrade, solution innovation, etc. can all be adopted/adapted cost-effectively.

2) Introduced a co-occurrence-based DSM approach for module partition based on the historical datasets in an objective way. The co-occurred models of existing data records are regarded as the source and target nodes respectively, while the co-occurrence frequency as the weights of the edges. Furthermore, the robustness of this transactional and continuous process is validated with the proposed principles.

3) Proposed a novel three-way based cost-sensitive learning approach for engineering change prediction. Three-way decision theoretical rough set together with random forest decision tree approaches are modified to be adapted in the existing dataset with the highest prediction accuracy (i.e. lowest average cost). Hence, a robust learning mechanism can be generated to predict/recommend future engineering changes under specific conditions.

To validate the proposed systematic approach, an illustrative example of a complex SCP is further introduced. As an explorative study, it is hoped that this research can provide insightful guide for manufacturer/service providers' engineering change management in the Smart PSS environment. On the contrary, this work has some limitations in its implementation scope: 1) due to its essence, the co-occurrence-based DSM cannot fully deal with cases that have much imprecise or incomplete information; 2) in this research, data sources in the condition dataset are deemed independent with each other, which might not always be true (e.g. riding speed and distance). Hence, it is recommended that future efforts can be also endeavoured in the following aspects: 1) introduce any enhanced algorithms (e.g. cost-sensitive learning) for the proposed approach with less computation time, complexity, etc.; 2) discover enabling ways (e.g. graph embedding) to cluster the heterogenous data sources in the condition set to map with decisions in the smart, connected environment; 3) any other effective methods other than co-occurrence-based approach for data-driven engineering change management.

Acknowledgement

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

Reference

- [1] D. Tang, R. Zhu, J. Tang, R. Xu, R. He, Product design knowledge management based on design structure matrix, *Adv. Eng. Informatics*. 24 (2010) 159–166. doi:10.1016/j.aei.2009.08.005.
- [2] L. Yin, D. Tang, I. Ullah, Q. Wang, H. Zhang, H. Zhu, Analyzing engineering change of aircraft assembly tooling considering both duration and resource consumption, *Adv. Eng. Informatics*. 33 (2017) 44–59. doi:10.1016/j.aei.2017.04.006.
- [3] N. Zhang, Y. Yang, Y. Zheng, J. Su, Module partition of complex mechanical products based on weighted complex networks, *J. Intell. Manuf.* (2017) 1–26. doi:10.1007/s10845-017-1367-6.
- [4] S. D. Eppinger, T. R. Browning, *Design structure matrix methods and applications*, MIT press. (2012).
- [5] D. V. Steward, The design structure system: A method for managing the design of complex systems, *IEEE Trans. Eng. Manag.* EM-28 (1981) 71–74. doi:10.1109/TEM.1981.6448589.
- [6] T.R. Browning, Design Structure Matrix Extensions and Innovations: A Survey and New Opportunities, *IEEE Trans. Eng. Manag.* 63 (2016) 27–52. doi:10.1109/TEM.2015.2491283.
- [7] M.E. Porter, J.E. Heppelmann, How Smart, Connected Product Are Transforming Competition, *Harv. Bus. Rev.* (2014) 64–89. doi:10.1017/CBO9781107415324.004.
- [8] M.E. Porter, J.E. Heppelmann, Connected Products Are Transforming Companies How Smart , Connected Products Are Transforming Companies, *Harv. Bus. Rev.* (2015) 1–9. doi:10.1017/CBO9781107415324.004.
- [9] P. Zheng, Y. Lin, C.-H. Chen, X. Xu, Smart, connected open architecture product: an IT-driven co-creation paradigm with lifecycle personalization concerns, *Int. J. Prod. Res.* 0 (2018) 1–14. doi:10.1080/00207543.2018.1530475.
- [10] P. Zheng, T.-J. Lin, C.-H. Chen, X. Xu, A systematic design approach for service innovation of smart product-service systems, *J. Clean. Prod.* 201 (2018) 657–667. doi:10.1016/j.jclepro.2018.08.101.
- [11] M. Goedkoop, *Product service systems*, Ecol. Econ. Basis. (1999).
- [12] F.H. Beuren, M.G. Gomes Ferreira, P.A. Cauchick Miguel, Product-service systems: A literature review on integrated products and services, *J. Clean. Prod.* 47 (2013) 222–231. doi:10.1016/j.jclepro.2012.12.028.
- [13] T.C. Kuo, M.L. Wang, The optimisation of maintenance service levels to support the product service system, *Int. J. Prod. Res.* 50 (2012) 6691–6708. doi:10.1080/00207543.2011.616916.
- [14] C. Lerch, M. Gotsch, Digitalized Product-Service Systems in Manufacturing Firms: A Case Study Analysis, *Res. Manag.* 58 (2015) 45–52. doi:10.5437/08956308X5805357.
- [15] S. Wiesner, E. Marilungo, K.-D. Thoben, Cyber-Physical Product-Service Systems – Challenges for Requirements Engineering, *Int. J. Autom. Technol.* 11 (2017) 17–28. doi:10.20965/ijat.2017.p0017.
- [16] A. Valencia, R. Mugge, J.P.L. Schoormans, H.N.J. Schifferstein, The design of smart product-service systems (PSSs): An exploration of design characteristics, *Int. J. Des.* 9 (2015) 13–28. doi:10.1016/j.procir.2016.04.078.

- [17] B. Kuhlenkötter, U. Wilkens, B. Bender, M. Abramovici, T. Süße, J. Göbel, M. Herzog, A. Hypki, K. Lenkenhoff, New Perspectives for Generating Smart PSS Solutions - Life Cycle, Methodologies and Transformation, *Procedia CIRP*. 64 (2017) 217–222. doi:10.1016/j.procir.2017.03.036.
- [18] P. Zheng, X. Xu, C.-H. Chen, A data-driven cyber-physical approach for personalised smart, connected product co-development in a cloud-based environment, *J. Intell. Manuf.* (2018) 1–16. doi:10.1007/s10845-018-1430-y.
- [19] A. McKay, S. Kundu, A representation scheme for digital product service system definitions, *Adv. Eng. Informatics*. 28 (2014) 479–498. doi:10.1016/j.aei.2014.07.004.
- [20] C. Lerch, M. Gotsch, Digitalized Product-Service Systems in Manufacturing Firms: A Case Study Analysis, *Res. Manag.* 58 (2015) 45–52. doi:10.5437/08956308X5805357.
- [21] J.B. Dahmus, G. Zugasti, K.N. Otto, Modular product architecture, *Des. Stud.* 22 (2001) 409–424. doi:10.1016/S0142-694X(01)00004-7.
- [22] Y. Li, X. Chu, D. Chu, Q. Liu, An integrated module partition approach for complex products and systems based on weighted complex networks, *Int. J. Prod. Res.* 52 (2014) 4608–4622. doi:10.1080/00207543.2013.879617.
- [23] F. Borjesson, K. Hölttä-Otto, Improved clustering algorithm for design structure matrix. In *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, (2012) 921-930. doi: 10.1115/DETC2012-70076
- [24] S. Jung, T.W. Simpson, New modularity indices for modularity assessment and clustering of product architecture, *J. Eng. Des.* 28 (2017) 1–22. doi:10.1080/09544828.2016.1252835.
- [25] J.A. Gopsill, C. Snider, C. McMahon, B. Hicks, Automatic generation of design structure matrices through the evolution of product models, *Artif. Intell. Eng. Des. Anal. Manuf. AIEDAM*. 30 (2016) 424–445. doi:10.1017/S0890060416000391.
- [26] L. Qiao, M. Efatmaneshnik, M. Ryan, S. Shoval, Product modular analysis with design structure matrix using a hybrid approach based on MDS and clustering, *J. Eng. Des.* 28 (2017) 433–456. doi:10.1080/09544828.2017.1325858.
- [27] J. Chen, S. Zhang, M. Wang, C. Xu, A novel change feature-based approach to predict the impact of current proposed engineering change, *Adv. Eng. Informatics*. 33 (2017) 132–143. doi:10.1016/j.aei.2017.06.002.
- [28] M.E.J. Newman, Modularity and community structure in networks, *Proc. Natl. Acad. Sci.* 103 (2006) 8577–8582. doi:10.1073/pnas.0601602103.
- [29] H. Cheng, X. Chu, A network-based assessment approach for change impacts on complex product, *J. Intell. Manuf.* 23 (2012) 1419–1431. doi:10.1007/s10845-010-0454-8.
- [30] X. Jin, S. Yu, P. Zheng, Q. Liu, X. Xu, Cloud-based approach for smart product personalization, *Procedia CIRP*. 72 (2018) 922–927. doi:10.1016/j.procir.2018.03.256.

- [31] S.K. Das, D.J. Cook, Smart Home Environments: A paradigm based on learning and prediction, *Wirel. Mob. Sens. Networks Technol. Appl. Futur. Dir.* (2006) 337–356.
- [32] J. Lee, H.A. Kao, S. Yang, Service innovation and smart analytics for Industry 4.0 and big data environment, *Procedia CIRP.* 16 (2014) 3–8. doi:10.1016/j.procir.2014.02.001.
- [33] V. Parida, D.R. Sjödin, J. Wincent, M. Kohtamäki, Mastering the transition to product-service provision: Insights into business models, Learning activities, and capabilities, *Res. Technol. Manag.* 57 (2014) 44–52. doi:10.5437/08956308X5703227.
- [34] E. Marilungo, A. Papetti, M. Germani, M. Peruzzini, From PSS to CPS Design: A Real Industrial Use Case Toward Industry 4.0, *Procedia CIRP.* 64 (2017) 357–362. doi:10.1016/j.procir.2017.03.007.
- [35] M. Grieves, Digital Twin: Manufacturing Excellence through Virtual Factory Replication, (2014) 1–7. http://innovate.fit.edu/plm/documents/doc_mgr/912/1411.0_Digital_Twin_White_Paper_Dr_Grieves.pdf.
- [36] P. Zheng, X. Xu, S.Q. Xie, A weighted interval rough number based method to determine relative importance ratings of customer requirements in QFD product planning, *J. Intell. Manuf.* (2016) 1–14. doi:10.1007/s10845-016-1224-z.
- [37] K. Guo, Y. Tang, P. Zhang, CSF: Crowdsourcing semantic fusion for heterogeneous media big data in the internet of things, *Inf. Fusion.* 37 (2017) 77–85. doi:10.1016/j.inffus.2017.01.008.
- [38] M.E.J. Newman, Analysis of weighted networks, *Phys. Rev. E - Stat. Physics, Plasmas, Fluids, Relat. Interdiscip. Top.* 70 (2004) 9. doi:10.1103/PhysRevE.70.056131.
- [39] V.D. Blondel, J.L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, *J. Stat. Mech. Theory Exp.* 2008 (2008) 1–12. doi:10.1088/1742-5468/2008/10/P10008.
- [40] Z. Pawlak, Rough sets, *Int. J. Comput. Inf. Sci.* 11 (1982) 341–356. doi:10.1007/BF01001956.
- [41] H.R. Zhang, F. Min, Three-way recommender systems based on random forests, *Knowledge-Based Syst.* 91 (2016) 275–286. doi:10.1016/j.knosys.2015.06.019.
- [42] A. a. Hagberg, D. a. Schult, P.J. Swart, Exploring network structure, dynamics, and function using NetworkX, *Proc. 7th Python Sci. Conf.* 836 (2008) 11--15. http://www.osti.gov/energycitations/product.biblio.jsp?osti_id=960616.