Service-driven approach to life-cycle dynamic evolutionary update of modular product structure

Lei Zhou1, Pinglu Chen1, Jing Xu1, Wen ting Qiu1, Muhua Liu1

1College of Engineering, Jiangxi Agricultural University, Nanchang, People’s Republic of China
E-mail: xujing0085@sina.com

Abstract: In recent years, product service system (PSS) has been paid more and more attention. Providing personalised products for customers is one of the most important services. The modularity based customisation is an effective means for it. For better customisation, the update of modular product structure (MPS) is necessary, which is the key to maintain the competitiveness of products. Therefore, the update approach of MPS for the life-cycle dynamic evolution is studied, and the problems that need to be solved emphatically are discussed. In view of the different objects concerned in the whole life cycle, a phased evolution system of MPS is constructed. Based on this, the driving factors, levels and ways of update are analysed accordingly. Then four analysis indicators are proposed to determine whether to update, and the technical supports of evolutionary update are illustrated. Finally, through the analysis of the process of dynamic evolution and update of micro-cultivator MPS, it shows that the dynamic update of MPS in the whole life cycle can maintain the ability of PSS to respond to market demand quickly, and lay a foundation for providing high-quality personalised service.

1 Introduction

The main goal of product service system (PSS) is providing personalised products and services to customers. PSS is an effective means to improve customer satisfaction and enterprise competitiveness. At present, a large number of scholars focus on the research of PSS. For the maintenance service, Chang et al. [1] attempted to formulate the multiplayer maintenance grouping strategy for complex multi-components system based on game theory. Hristo et al. [2] proposed a PSS integration modelling framework based on system modelling language. Li et al. [3, 4] proposed a corresponding PSS development framework based on modular design in order to realise rapid mass customisation of personalised products. Wallin et al. [5] studied how manufacturing enterprises establish the innovation ability of PSS, and explored the importance of innovation ability to the development of PSS. Mourtzis et al. [6] pointed out that currently there are few studies on the integration and evaluation of PSS design, so they proposed a framework to evaluate and improve the lean PSS design with key performance indicator (KPIs), lean rules and sentiment analysis, considering each stage of the PSS design lifecycle. Chang et al. [7] proposed a function availability-based integrated product-service network model for high-end manufacturing equipment (HEME), which can better ensure functional availability by linking to more critical integration services.

Personalised products are the basis to meet customer needs. Providing personalised customised products is also one of the most basic services of enterprises, and researches on personalised product customisation are also quite wide [8]. Personalised product customisation is often based on reuse design and modular design. Wei and Shao [9] combined cloud manufacturing technology to conduct personalised product customisation, and Dou et al. [10] searched for the optimal solution of personalised product customisation through multi-stage interactive genetic algorithm. Most of the existing researches in the field of personalised product customisation focus on the configuration method, in order to get the personalised product by module matching according to customer needs. However, there are few researches on the update of the modular product structure (MPS). Cavalcante and Gizara [11] point out that there are considerable application prospects for improving the provision and related activities of PSS through lifecycle information, so it is necessary to update the MPS from a lifecycle perspective in order to better provide personalised product customisation services.

MPS is not once and for all, without update, the internal modules and other things are unchanged, and it will be difficult to adapt to the future customer needs, more cannot achieve the effect of rapid configuration. Some scholars have studied the classification of parts. The establishment of classification structure is conducive to the renewal [12, 13]. Xu et al. [14] studied on the classification and coding of parts for the purpose of updating new parts. The research of corresponding data management method has laid a foundation for the update of MPS. Arcaini et al. [15] proposed an evolutionary approach to feature model changes in SPL engineering area. SCH-TTNER [16] gave a detailed explanation of product data management in manufacturing enterprises, Jiang and Li [17] designed digital industrial products based on the product data management system and Lin [18] proposed a product data management system architecture for small and medium enterprises. Qi et al. [19] managed modular product data from the perspective of product master structure, and discussed the storage method of data such as module, component related master document and master model.

The original intention that we propose the evolutionary research is to continuously provide customers with personalised product customisation services. To continuously provide these services, the update of MPS is necessary. New products and modules will appear in the process of customisation service, and the relevant data in the process such as maintenance service will be used as the basis for variation and improvement design, leading to the birth of new modules, which will promote the update. Therefore, the ultimate goal of evolutionary update is to provide better services, which is complementary to services.

2 Lifecycle dynamic evolutionary update system of MPS

In order to maintain the competitiveness of MPS through evolutionary update, it is first necessary to establish a whole lifecycle evolution model based on its internal modules, parts and various documents to provide different data for different stages of the life cycle [20]. Evolutionary update is to change the evolution model. Due to the different information paid attention to the development, design, processing and assembly, sales and service
stages in the life cycle, the MPS needs to be decomposed into a tree structure applicable to each stage when construct the whole life-cycle evolution model, in order to make the evolution model more detailed. Since each stage of life cycle focuses on different information such as the function requirements for development stage, the specific structure design of function for design stage and the control of parts size and technology for production stage, in our previous studies, we had decomposed the MPS into the function tree in the development stage, the principle and structure tree in the design stage and the processing and assembly tree in the production stage.

The nodes of the function tree are independent functions defined one by one, which are called function modules. Each function module contains attribute information such as function name, function ID, function description and so on. The nodes of the principle and structure tree are divided into two categories, and one is the principle solution node, the other is the structure entity node, which are called the principle solution module and the structure entity module, respectively. Principle solution refers to a way to realise a certain function, such as chain transmission and gear transmission are two different principle solutions. Structure entity is an exact structure of principle solution, for example, different transmission ratios of the gear transmission are different structure entities for principle solution of gear transmission. In addition to the attribute information such as name, ID, description, the principle solution module also contains parameterised 3D models. Based on the 3D model of principle solution, the structure entity module determines specific parameters to obtain specific 3D models and generate 2D drawings. The main nodes of processing and assembly tree are structure entity nodes and part nodes, in which part nodes refer to the set of component parts of each structure entity. Parts also have name, ID, description and other attribute information, and the main model should be established for each type of part. The 3D model and 2D drawings of each specific part are derived from the main model. Building the corresponding processing documents for parts is also necessary.

There are parent–child, composability relationship and subordination relationship are used to define which other types of objects an object contains, such as which principle solutions a function exists, and which structure entities of a principle solution exist. These relationships can be expressed separately in the form of matrix, which use the object's ID as the column and row of the matrix, and the 0, −1, 1 and so on values as the matrix elements to represent the relationship between the two objects. For example, in the parent–child relationship matrix of functions, 0 means that there is no parent–child relationship between functions, 1 means that the corresponding function of the row is the parent function of the corresponding function of the column and −1 is the opposite; and the value ‘2’ can be added to the composability relationship matrix of principle solutions, where 1 means that the two principle solutions can be combined with simple modification and so on.

In order to better implement the whole life cycle evolutionary update of the MPS, the evolutionary update framework is proposed as shown in Fig. 1. Based on this framework, the evolutionary update process will be analysed and guided. Evolutionary update is the process of transforming the old evolutionary model into the new one. Therefore, it is necessary to understand:

(i) What drives the update;
(ii) What needs to be updated;
(iii) What are the forms to update;
(iv) How to determine whether update is needed;
(v) How to implement updates.

3 Lifecycle dynamic evolutionary update method of MPS

3.1 Driving factors of MPS life-cycle dynamic evolutionary update

Evolutionary update will not be carried out without any reason, because updating the evolutionary model means that the enterprise needs to invest corresponding capital, and random evolutionary update will inevitably cause unnecessary cost. Therefore, only by understanding the driving factors of evolutionary update can we guide the evolution model to make reasonable changes and maintain the vitality of the MPS.

The MPS is constantly updated to adapt to the ever-changing market environment to maintain its vitality, and the market environment contains many aspects such as government policies,
customer needs, the status of relevant supply chains and so on. In addition, evolutionary updates may also come from the internal of enterprise. In order to better manage the MPS, enterprises often need to optimise the composition structure of the MPS. These internal and external factors are the main driving forces of evolutionary update. Evolutionary update can be divided into active update and passive update according to the mandatory degree of driving factors act on MPS.

3.1.1 Passive update (high mandatory degree): When the new customer needs or government mandatory policies are coming, the enterprises cannot meet these demands and polices by the existing modules and components, the passive update must be done. Only through the new design or variant design of the corresponding modules and parts can the enterprise meet the new demands. After the new modules and parts generated, they should be evaluated to determine whether they can update into the evolution model and how to update them.

3.1.2 Active update (low mandatory degree): Even though there is no new customer demands or new mandatory government policies, the enterprises could also update their modules and parts voluntarily according to future possible customer needs and government policies they predicting, and this update process is active update. Active update can also be made when the old modules and parts cannot meet the management requirements of evolution model.

3.2 Hierarchy of MPS life-cycle dynamic evolutionary update

Since the evolution model is based on modules and parts, and the process of evolution update is the process of updating modules and parts, including updating the module and parts’ attribute and relationship information. According to the different scope involved in the update process, evolutionary update can be divided into two aspects: module update and part update. As modules are defined in stages as the function module, principle solution module and structure entity module as mentioned above, the update of module level can also be subdivided into three levels: function module update, principle solution module update and structure entity module update.

3.2.1 Part update: When existing parts are not enough to constitute a new structure entity, and the processing technology and structure are backward and so on, new parts should be designed and old parts should be removed or do variant design, so does the information of parts in the part library, in order to make the evolution model updating in the level of part.

3.2.2 Module update: In this paper, modules are divided into three different types, and they have a subordinate relationship, among which the function module has the highest level, the principle solution module has the second level and the structure entity module has the lowest level. It means that a function module can have multiple principle solution modules, and a principle solution module can have multiple structure entity modules. The update of modules at different levels will affect each other. The update of modules at the higher level may cause the update of modules at the lower level, and the update at the lower level may also cause the update at the higher level. At the same time, there is a parent–child relationship between modules of the same type. The update of the parent module and the child module of the same kind will also affect each other. When the parent module is updated, whether the child module is updated with it or not should be considered (as shown in Fig. 2).

(i) Update of high-level modules: When there is a new demand in the market, and there is no function module corresponding to the demand in the existing evolution model, it is necessary to update the evolution model. At this moment, the first thing to be updated is the function module. The developer needs to accurately define the new function module based on the market demand and propose the design index of the function; secondly, with the update of the function module, the principle solution module needs to be updated accordingly, to provide different ways to realise the function.

(ii) Update of low-level modules: If an order can be realised relying on the existing function module and the principle solution module, but cannot meet the performance requirements, the existing structure entity needs to be improved. If the improved structure entity module wants to be updated into the evolution model, it only needs to update the subordinate relationship between the new structure entity module and the old principle solution module, while the principle solution module does not need more changes in general. However, if the enterprise has designed a new structure for a product order that does not belong to any existing principle solution module, then when it is updated as a structural entity module, it still needs to update the principle solution module. Which means enterprises need to add the principle solution module.

---

**Fig. 2 Transmission of evolution updates at the module level**

![Diagram showing the transmission of evolution updates at the module level](http://creativecommons.org/licenses/by/3.0/)

This is an open access article published by the IET under the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/)
containing the structure entity, or even the higher-level function module. Therefore, the update of low-level modules needs to be analysed according to the specific situation to determine the update of high-level modules.

Therefore, whether it is a part-level or module-level object, as long as there is any relationship between the two objects, when updating one of the objects, we need to consider whether the other object needs to be updated. It needs to be analysed according to the specific situation (as shown in Fig. 2).

3.3 Forms of MPS life-cycle dynamic evolutionary update

No matter what level of evolutionary update can be divided into four ways: addition, elimination, modification and replacement.

3.3.1 Addition: It means that new modules or parts are added to an evolution model. Due to market changes, existing evolutionary models cannot continuously meet changing needs. Designers will constantly design new parts and modules. After the relevant evaluation process and making a decision to update them into the evolution model, the attribute and relationship information of them need to be defined, then added them into the corresponding positions of the tree structure of each stage in the form of nodes.

The corresponding edges should be added to determine the parent–child relationship (as shown in Fig. 3).

For example, adding a function module. First of all, the node position of the function module in the function tree should be determined. The relevant description and coding should be established for the function, the drawings and models are unnecessary to add. Meanwhile, we also need to consider whether to modify its parent function or add child functions to the function. Then, it is necessary to add the principle solution modules for the series of function modules added newly, and add the respective structure entity modules to corresponding module. Meanwhile, not only the relevant attribute information of each module need to be added, the relationship information among modules also needs to be modified accordingly. If some parts of the new structure entity are not included in the evolution model, then we also need to add these parts to the evolution model.

3.3.2 Elimination: It refers to the removal of some modules or parts in the evolution model that do not meet the current market demand. It means that the relevant nodes and information are removed from the tree of each stage in the evolution model (as shown in Fig. 4).
For example, when a function module is eliminated, its corresponding child-function module also needs to be eliminated and its parent function module needs to be modified. At the same time, the relevant principle solution module and structure entity module need to be removed from the evolution model. Meanwhile, the relevant attribute information of each module needs to be deleted, and the relationships between other module and the eliminated module also need to be deleted. If a part needs to be eliminated, the structure entity composed of the part should be considered to be deleted or not.

3.3.3 Modification: It means that minor modification of relevant modules and parts in the evolution model, such as the modification of relevant descriptive documents for the requirements of management. Modifications often do not cause other existing modules to crash. For example, redefining the name of a structure entity does not invalidate the parent structure entity which was composed of this structure entity.

3.3.4 Replacement: It means that a part or module needs to be eliminated, and a new part or module is added to replace it. Replacement often occurs when the original parts, modules need a relatively large modification, and the modified parts or modules are huge different to the original one. They can be regarded as new ones, but occupy the unique code of the original parts or modules.

In a word, it is necessary to analyse whether the related module or part need to update when a module or part is added, eliminated, modified or replaced.

3.4 Analysis indicators to judge whether MPS life-cycle dynamic evolutionary updates should be performed

Evaluation criteria are needed for life-cycle dynamic evolutionary update of MPS. For new parts and modules, update is taken directly to the evolution model for subsequent reuse. However, if there are some structure never seen, it cannot be directly added into evolution model, the module division need to be done first [21] (refer to Fig. 3). If there is a need for management, the minor modifications are also often taken directly to the relevant parts or modules. However, both elimination and replacement need long-term observation, then determine whether they are implemented according to the data observed (refer to Fig. 4). Therefore, the following four types of analysis indicators are proposed.

3.4.1 Reuse degree \( f_i \): It represents the reuse of a part or module over a period of time. A high reuse degree indicates a larger production batch

\[
f_i = \frac{n_i}{N}
\]

where \( N \) represents the total output of the enterprise in the recent period, \( n_i \) represents the number of products that have reused module \( i \) and \( f_i \) represents the reuse frequency of module \( i \) (see (1)).

3.4.2 Customers satisfaction degree \( S_{ij} \): It represents the customer's evaluation of a product or module; if the satisfaction is too low, the module should be eliminated. Evaluation indicators are established from multiple aspects, each aspect has respective scoring standards and weigh. Customers are required to participate in the evaluation in the form of after-sales feedback

\[
S_{ij} = \sum_{j=1}^{m} \sum_{i=1}^{n} w_i p_{ij} m
\]

The \( n \) and \( m \) represent the number of indicators and customers, respectively, \( w_i \) is the weight of indicator \( i \), \( p_{ij} \) represents the score of the module given by customer \( j \) on indicator \( i \) (see (2)).

3.4.3 Profit rate \( (P_i) \): It represents the ratio of profit and cost of a module. If the profit rate of this module is too low, it should be eliminated

\[
P_i = \frac{P_i}{C_i}
\]

The \( P \) is the profit of the module, which is obtained by subtracting the selling price from the cost. \( C \) is the cost of the module (see (3)).

3.4.4 Stability \( (S_t) \): It refers to the failure of the module. High failure rate will directly affect customer satisfaction. Continuous return to the factory for maintenance will also lead to a decrease in profit. Ultimately, reuse become more and more difficult

\[
S_t = \frac{n_t}{N_t}
\]

where \( N_t \) represents the number of times a product has worked in the recent period, \( n_t \) represents the number of times a module has failed (see (4)). Of course, this equation is only a simple calculation, the more specific evaluation should also consider the scale of the fault, the maintain cycle and so on.

According to the above indicators, we can determine whether to eliminate. As for replacement, it is necessary to consider whether the above indicators are increased when the new modules are compared with the old ones.

However, these indicators are mainly provided for entitative elements such as structure entity and parts. As for principle solution, those indicators can be evaluated by the structure entities it contains. However, because function modules are abstract, so those indicators do not apply to function modules. It is worth mentioning that these indicators are not absolute indicators, we cannot give an exact value to indicate that we should perform an elimination or replacement operation when the indicator reaches this value. These indicators are just to provide a numerical representation for expressing the various considerations of the enterprise. On this basis, enterprise personnel can combine their own management needs, product customisation needs and other aspects to carry out a comprehensive analysis.

For example, there are two principle solutions A and B that can achieve the same function. The reuse degree of A is higher than B, but we cannot eliminate B just by this indicator, because B may have higher stability or profit rate. The reason for its low reuse degree may be that its product positioning is relatively high, which is different from the customer groups targeted by A. Therefore, these indicators proposed in this paper should be used as auxiliary indicators for enterprise decision making. Whether to update or not still requires the enterprise to combine with its own product characteristics to carry out a comprehensive consideration of various factors.

3.5 Technical implementation of MPS life-cycle dynamic evolutionary update

Above, the driving factors, levels, forms and implementation analysis indicators of evolutionary update are described. However, these belong to the discussion at the level of analysis and decision. In addition, the key issue is how to implement the update, that is, the technical support of evolutionary update. As mentioned above, evolutionary update is the update of the evolution model, and also the update of tree nodes and edges in each stage. Therefore, the essence of evolutionary update is to update the attributes of each object and the relationship between objects. Redefining the object by changing the attribute information of each object, and searching for other objects that may be updated with the current object being updated by the parent–child relationship, subordination relationship and composability relationship. While updating the attribute information of the object, the relationship between each object also needs to be updated. For example, the parent–child relationship between the new principle solution object and the existing principle solution needs to be added to the parent–child relationship matrix.
of principle solution, and the subordination relationship between the new principle solution and the existing function needs to be added to the subordinate relationship matrix between function and principle solution.

The technical implementation process of evolution update is shown in Fig. 5. The function and the principle solution object are used as examples for evolutionary update demonstration in the figure, and the execution of other object is the same. There is a function that needs to be updated (called the current function), which is the starting point of the update. Next is step 1, to determine whether to update and in what form; then go to step 2, to update the attributes of this object and various relationships with other objects; next, step 3 and step 4 are used to automatically link the objects that need to be updated according to each relationship matrix; then go to step 5 and update the objects that may need to be updated. Loop steps 1–5 until there are no objects that need to be updated. Through the above methods, the evolution prototype system can be developed to manage the update process.

4 Application

In this paper, a modularised product structure of an agricultural micro-cultivator is taken as an example to demonstrate and analyse the evolutionary update. Firstly, the MPS evolution model of agricultural micro-cultivator is established (Fig. 6).

The MPS evolution update case of micro-cultivator is shown in Fig. 6, the main function of micro-cultivator is ‘Till Land Function’, and one child function of it is ‘Walking Transmission Function’. In this paper, for the sake of diversity, the walking transmission function is divided into two child functions, the ‘Walking Transmission Function (upper)’ and ‘Walking Transmission Function (lower)’, so we can obtain as many kinds of micro-cultivator as possible by matching the corresponding structure entities of different child functions. The solid line in the figure represents the existing functions, principle solutions and structure entities. For example, the ‘Walking Transmission Function (upper)’ has the principle solution of ‘Front:2-gear/Back:1-gear/Straight teeth gear shift/Bevel gear output’, while the ‘Walking Transmission Function (lower)’ has the principle solution of ‘2-Staged Bevel Gear Transmission’, and there are corresponding structure entities for them, respectively, which is shown by the 3D model in Fig. 6.
has designed a new form of principle solution for the two function, the ‘Front:2-gear/Straight teeth gear shift/Chain wheel output’ and ‘1-Staged Chain Wheel Transmission’, then design the structure entities for them. At the same time, the parent functions, ‘Walking Transmission Function’, also appeared the new principle solution which is called ‘Front:2-gear/Straight teeth gear shift/Chain wheel output + 1-Staged Chain Wheel Transmission’. The structure entity of this new principle solution is composed by the structure entities of its child principle solutions. Since this update does not involve the addition of functions, the function module does not need to be updated. What is involved in the update is contained in the dashed box in Fig. 6. The other forms of update operations are same to this, we need to update the affected modules and parts according to the scale of influence.

Taking the principle solution as an object, we present two cases to demonstrate how evolutionary updates are implemented.

As shown in Fig. 7, when there is a new principle solution needs to be updated, it first needs to be located to the function it subordinates to through a level-by-level selection. When a certain level of function is selected, the inferior list will automatically list its corresponding child functions and the details of child functions. If no existing function can correspond to the new principle solution, the new function needs to be added; otherwise, the list of existing principle solutions of the current function is displayed so that similar principle solutions can be selected as update templates for the new principle solution. After selecting the template, enter the corresponding attribute information for new principle solution and update the various relationship matrices. Then determine whether to add the parent, child principle solutions and structure entities, and perform the interactive update.

As shown in Fig. 8, for the update of current principle solution, the four types of analysis indicators mentioned in Section 3.4 should be calculated first, and then the update forms, modification or elimination or replacement should be decided according to the results. Then based on the parent–child relationship, subordination relationship and composability relationship, the function, structure entities and parent or child principle solution and so on will be listed automatically, which need to be updated on demand.

5 Conclusion
Considering different stages in the whole life cycle, the function module, principle solution module and structure entity module for different stages are proposed, and the evolutionary update model is constructed. By updating the parts and different types of modules, the evolution of the whole MPS is realised. There are four forms to realise the evolutionary update of parts and modules: addition, elimination, modification and replacement. Which form to update should be analysed according to the specific situation. In addition, evolutionary updates occur at different levels, and updates at each level will directly or indirectly affect updates at other levels. It is necessary to start from the update at the current level to complete the update of corresponding objects, otherwise it will affect the ability to provide personalised products through the evolution model in the future. Now, according to the implement method of evolutionary update at the technical level, we are still developing the evolutionary prototype system, and hope that the effective management of the update process can be achieved with this system.

6 Acknowledgments
This project is supported by the National Natural Science Foundation, China (no. 31660349).

7 References
[1] Chang, F.T., Zhou, G.H., Cheng, W., et al.: ‘A service-oriented multi-player maintenance grouping strategy for complex multi-component system based on game theory’, Adv. Eng. Inf., 2019, 42, p. 100970
[2] Hristo, A., Matthias, F.: ‘Modeling framework for integrated, model-based development of product-service systems’, Procedia CIRP, 2018, 73, pp. 9–14

[3] Li, H., Tao, F., Wen, X.Y., et al.: ‘Modular design of product-service systems oriented to mass personalization’, China Mech. Eng., 2018, 29, (18), pp. 2204–2214, 2249

[4] Li, H., Ji, Y.J., Chen, L., et al.: ‘Bi-level coordinated configuration optimization for product-service system modular design’, IEEE Trans. Syst., Man, Cybern.: Syst., 2017, 47, (3), pp. 537–554

[5] Wallin, J., Parida, V., Isaksson, O.: ‘Understanding product-service system innovation capabilities development for manufacturing companies’, J. Manuf. Technol. Manag., 2015, 26, (5), pp. 763–787

[6] Mourtzis, D., Fotia, S., Vlachou, E., et al.: ‘A lean PSS design and evaluation framework supported by KPI monitoring and context sensitivity tools’, Int. J. Adv. Manuf. Technol., 2018, 94, pp. 1623–1637

[7] Chang, F.T., Zhou, G.H., Xiao, X.P., et al.: ‘A function availability-based integrated product-service network model for high-end manufacturing equipment’, Comput. Ind. Eng., 2018, 126, pp. 302–316

[8] Wang, Y., Ma, H.S., Yang, J.H., et al.: ‘Industry 4.0: a way from mass customization to mass personalization production’, Adv. Manuf., 2017, 5, (4), pp. 311–320

[9] Wei, H.J., Shao, B.L.: ‘Study of product personalized customization based on cloud manufacturing’, Chin. J. Eng. Des., 2018

[10] Dou, R., Zong, C., Nan, G.: ‘Multi-stage interactive genetic algorithm for collaborative product customization’, Knowl.-Based Syst., 2015, 92, pp. 43-54

[11] Cavalcante, J., Gzara, L.: ‘Product-service system lifecycle models: literature review and new proposition’, Procedia CIRP, 2018, 73, pp. 32–38

[12] Huang, R., Zhang, S., Bai, X., et al.: ‘Multi-level structuralized model-based definition model based on machining features for manufacturing reuse of mechanical parts’, Int. J. Adv. Manuf. Technol., 2014, 75, (5–8), pp. 1035–1048

[13] Braglia, M., Grassi, A., Montanari, R.: ‘Multi-attribute classification method for spare parts inventory management’, J. Qual. Maint. Eng., 2004, 10, (1), pp. 55–65

[14] Xu, X., Ji, Y.J., Qi, G.N., et al.: ‘Classification and coding method of mechanical parts for the design process of mass customization’, J. Mech. Eng., 2010, 46, (11), pp. 149–155

[15] Arcaini, P., Gargantini, A., Radavelli, M.: ‘Achieving change requirements of feature models by an evolutionary approach’, J. Syst. Softw., 2019, 150, pp. 64–76

[16] Schütter, J.: ‘Product data management in manufacturing industry’ Translated by Qi Guoning, (China Machine Press, Beijing, 2000)

[17] Jiang, X.F., Li, J.Y.: ‘Research on the digital industrial design system integrated with product data management system’, Adv. Mater. Res., 2010, 102–104, pp. 106–110

[18] Lin, H.: ‘The design and implementation of PDM system for small and Medium enterprises’, Appl. Mech. Mater., 2014, 607, pp. 826-829

[19] Qi, G.N.: ‘Graphical product data management’ (China Machine Press, Beijing, 2000)

[20] Xu, J., Zhou, L., Chen, P.L., et al.: ‘The evolutionary research of modular master structure oriented to product-service systems’, Procedia CIRP, 2019, 83, pp. 579–583

[21] Xu, J., Zhou, L., Chen, P.L., et al.: ‘Module granularity optimization analysis model based on digital twin’, Comput. Integr. Manuf. Syst., 2019, 25, (6), pp. 1419–1431