Framework and CORBA Implementation of A New Industrialized PL-ISEE Database Platform

Jianli Dong
Huaihai Institute of Technology / School of Computer Engineering, Lianyungang, 222005, P.R.China.
dongjl1019@sina.com

***Abstract*—Using the automated production and management system of modern manufacturing industry production line for reference, a new industrialized PL-ISEE model was proposed by authors in article [8]. In the new model, the middle part of the PL-ISEE architecture, data layer, is composed of the software components bus and environment database supporting platform which achieves the storage and managements of product line core assets data. According to the organisation structure and assembling needs of product line core asset components, the PL-ISEE database platform framework and its schema and view design as well as implementation methods are further researched in the paper. The data model, database schema and application views of the database platform are designed and created by using a united core asset data model and product line assembling evolution model. The schema and views based the database platform can well satisfy all the ability requirements of the core asset data integration and assembling producing mode of software product line engineering. Finally, the realizing method of the new industrialized PL-ISEE database support platform based on the CORBA architecture is also proposed and discussed in detail.**

***Index Terms*—software product line, PL-ISEE (product line based integrated software engineering environment), software architecture, software components bus, core assets, database schema, database platform**

I. INTRODUCTION

In recent years, with the growing maturity and application of new technologies such as software architecture, software component, large-granularity software reuse and so on, software engineering methodology based on product line, which rises the widespread concern and attention, has become a hot and key topic in software engineering field. The core of research on software product line is, like to the automated assembly line of the modern manufacturing industry products (such as cars, television), to implement the automated and industrialized mass-customized production of domain specific software products by the software architecture, software component and large-granularity system-level reuse techniques. This is the most ideal way of software production in the 40 years software engineering development, which would play a very important role for promoting the development and formation of modern software industry, and would result in enormous social and economic benefits [1-3].

The researches of the software engineering based on product line mainly focus on the product line engineering methodology, engineering processes and life-cycle model, PL-ISEE (product line based integrated software engineering environment), software assembly line, the software architecture and components, DSSA(domain specific software architecture), domain and application engineering, core assets and its database system, standardization and so on aspects. The paper mainly researches and discusses on the architecture and schema as well as view design of the database supporting platform of a new industrialized PL-ISEE proposed in reference [8].

II. RESEARCH AND RELATED PROBLEMS ON SOFTWARE PRODUCT LINE ENGINEERING

Compared to the traditional software development methods, the software engineering methodology based on product line would make fundamental changes on software development methods and techniques. Software development would gradually be transformed into the industrialization production of "software architecture + components + assembly line" with the producing systems and features of modern manufacturing industry (such as cars, television) from the traditional one-off and starting from ground zero manual programming way of "algorithms + data structures +manual programming". The implementation of software product line production methods must depend on the supporting of product line integrated development environment with the product line engineering features and producing capacities. This kind of supporting environment is known as PL-ISEE (product line based integrated software engineering environment). Therefore, the research, implementation and application of PL-ISEE have very important role in promoting the automation and industrialization of software industry, which have been the strategic initiatives for countries in the world to seize the information industry and promote the rapid and sustainable development of economy and society [4-5].

But, we should clearly understand that the formation and development of software product line engineering methodology are on the basis of domain engineering, software architecture, software components and software reuse techniques and completely use the experience of the industrialized producing methods and ideas of

Corresponding author:Jianli DONG, E-mail: dongjl1019@sian.com

doi:10.4304/jsw.7.7.1677-1686

modern manufacturing industry product line to build software product lines aimed at reusing the DSSA(domain specific software architecture) and system-level large-granularity components. The final object is to achieve industrialization and automation production of software products. Certainly, the study on the PL-ISEE is just to build an ideal supporting platform or environment for the software's assembling production. Clearly, there are essentially different between the PL-ISEE model and traditional all-purpose ISEE based on structured and object-oriented software engineering methodology. The former achieves the mass customization industrialization and automatic production the domain specific software products according to the method of "domain specific software architecture + components + assembly"; the latter achieves one-off and from scratch software development in accordance with the developing way of "data structure + algorithm + manual programming", and its commonality also determines that the environment's suitability, production mode, productivity, update and transformation would be subjected to many restrictions and differ with producing mode and development direction of the modern manufacturing industry. The reasons are the same as the modern manufacturing industry can not use a common assembly line to achieve the products production in different fields such as cars, airplanes, television and so on.

Unfortunately, the research and development of current PL-ISEE is still no clear understanding of the essential differences between it and traditional ISEE. It is not correct to apply traditional software engineering approaches and ideas in developing software engineering methodologies based on product line and integrated development environment. As shown in CMU/SEI (Carnegie Mellon University/Software Engineering Institute) and reference [6-7], there is not a true sense of PL-ISEE so far in the researches and developments of modern ISEE. In other words, the actual PL-ISEE research and development status are that the so-called PL-ISEE are formed by some software companies appropriately introducing the concept of software components and relative control facilities into their original software developing environment. The developing environment such as Rational of IBM, J2EE of SUN, etc. are all belong to this kind of type, and also far below the true sense of PL-ISEE with assembling line production capacities and features like modern manufacturing industries. Of course, analysis from the benefits of software enterprises and their decades of development modes, especially system and tools software development processes, this approach is natural and understandable because companies must take the existing products and benefits into account. Despite of the wide range of applications and huge economic benefits, it is not impossible to abandon or overthrow the current large-scale and widely applied software products overnight completely to pursue the new products with regardless of the risk.

So, using the correct idea and way of modern manufacturing industry product line for reference, we firstly propose a new industrialized PL-ISEE model with software component bus in reference[8]. In this paper, according to the new PL-ISEE model, we will further research on the schema and views of the PL-ISEE database platform and their implementation mechanism based on the CORBA architecture, and hope to open a new approach for realizing software's industrialized assembling production with the core asset components of product line.

III. THE ARCHITECTURE OF A NEW PL-ISEE AND ITS DATABASE SUPPORTING PLATFORM

A. A New Industrialized PL-ISEE Model

We have proposed a new industrialized PL-ISEE model which is shown as the Figure 1 in reference [8]. This new industrialized PL-ISEE is multi-layer architecture based on the unified product line engineering concept model, large-granularity reusable core assets data model, component assembly behavior model, core assets development and application software production iterated evolution model [9]. The whole PL-ISEE architecture is composed of three layers of interface, tool and data layer.

The framework of this new industrialized PL-ISEE is essentially a hiberarchy double-development environment with core asset components as the software components bus. On the bus (or data layer), it is the assembly line of the PL-ISEE based on product line software engineering methodology, its function is to realizes the automated and industrialized assembling production of application software product clusters, and all the asset components needed by assembly line are provided by the software component bus in the producing process of software product line. Under the bus (or data layer), it is the traditional ISEE (integrated software engineering environment) based on traditional software engineering methods, its function is to support the development of component programs and documents of product line core assets. The middle part in the framework is data layer which consists of product line core asset component bus and environment database support platform, and its function is to achieve the data integration and management of the PL-ISEE as well as transmit the core asset components onto software assembly line as a conveyor belt.

In the reference [8], the PL-ISEE architecture model mainly consists of the three layers of interface layer, tool layer and data layer. Due to this paper mainly research on the architecture and implementation mechanism of the PL-ISEE database supporting platform in the data layer, so interface and tool layers are discussed and studied in other papers.

The functions and configurations of environment data layer mainly realize the integration, storage, sharing and management of product line environment data. The data layer includes the core asset and agent component bus and environment database supporting platform. The basic

architecture and realizing mechanism of the environment database platform will be discussed below.

B. OODB Requirements of PL-ISEE Database Supporting Platform

The architecture and design problem of PL-LSEE database platform is especially emphasized during the research of PL-ISEE data integrating implementation mechanism. Core assets of product line mainly contains product line software architecture, components, connectors, production projects, development documents, test projects and use cases, standards and specifications as well as many other complicated heterogeneous data assets. Thus, complicated heterogeneous core assets data model and core assets database schema, architecture and design of core assets database platform are the keys to achieve environment data integration and management. About the design of database model, one of the most widespread models is relationship database model recently. Since the capability defects of relationship model, such as expression ability and semantic gap, the design of product line core assets data model needs an object-oriented database model with perfect expression ability and well operation mechanism, in order to finish the expression and modeling requirements of complicated heterogeneous core assets data. On the design of core assets database schema, we research and create the multi-view core assets database schema with not only the three-level organization structure of product line architecture, architecture frameworks and components, but also mapping view, reuse view and relationship view, to archive the database ability of product line engineering features and management. For the implementation of core assets database support platform, management functions as storage, classification, index, optimization, version and configuration and so on are not only provided, but wonderful mechanism and interfaces with data integrating, tools integrating and interface integrating as well [10-12].

Based on the ability demand of PL-LSEE database platform above, PL-LSEE database system must satisfy the following functions and performance requirements: modeling ability of complex data object, profuse semantic model, dynamic schema evolution, strait constrains management, mass data storage, modeling and expression of meta data, data sharing, client communication, flexible transaction structure, data accessing and recovery, computing integrated database program design language, management of data integrity and consistency, data extension and integration, visual developing environment, and version and configuration management.

However, these function and performance requirements of PL-ISEE database platform can not be satisfied by traditional RDBMS (relation database management system), which has already been proofed through researches and practices. According to this, we must focus on the OODB (object oriented database) with engineering complex data expression, storage and management, since OODB have good mechanism and abilities to satisfy the requirements of PL-ISEE database platform. So, it is a natural and inevitable selection that chooses OODB as PL-ISEE database system, and also the fundamental motivation of researching and developing OODB. On the other hand, the requirements from PL-ISEE to OODB indicate that the whole PL-ISEE architecture and implementation will hold a united OO (Object Oriented) concept model and OO data model which completely adapted to OODB, which is very important to guarantee the cooperation, communication and data share among all the elements (such as data, interfaces and tools) integrated in PL-ISEE.

C. The PL-ISEE Database Platform Architecture

The architecture of PL-ISEE database platform consists of AOMS (asset objects management system) and EDBS (environment database system) as an hiberarchy shown in Figure 1. The function of the database platform is effectively to support the integration of the environment tools and data. So, AOMS must provide the perfect mechanism, united data model and interface for realizing the integration of enviroment tools, data and interfaces; EDBS provide the storage and management ability of product line core asset data objects.

As the architecture and implementation of EDBS, we use a united object oriented concept model and data model as the database model of PL-ISEE environment. This model makes the environment data's modeling, expression, storage and management much effectively. As AOMS must provide a general data interface for tools integrated on environment data base platform and some relative database aided operation tools for the EDBS. Fortunately, current OODBMS (object oriented data base management system) use OOPL (object-oriented programming language) integrated with OODB seamlessly as ODL (data description language) and OML (object manipulation language). This makes the all integrated tools own the same "OO thought" and talk "OO language (OOPL)". It is definitely the same as the object model and object interface provided by object-oriented environment database. So, all the environment tools that will access and share OODB can conveniently implement the integration of tool and data as well as other effective operations by using OODB inherent OO interface, and without any works on interface to achieve the "plug and play" result on the environment database platform. According to this, the highlight of platform research is to provide various kinds of database aided operation tools.

The design and architecture of PL-ISEE environment database system are consisted of several core asset sub-bases: RA-Dbase (product line requirement analysis assets database), A-Dbase (product line architecture assets database), F-Dbase(product line framework assets database), C-Dbase (product line components assets database which includes two kinds of SC-Dbase (source code components base) and EC-Dbase (executable code components base)), T-Dbase (product line test assets database), CM-Dbase(product line configuration management assets database) and so on. Above product

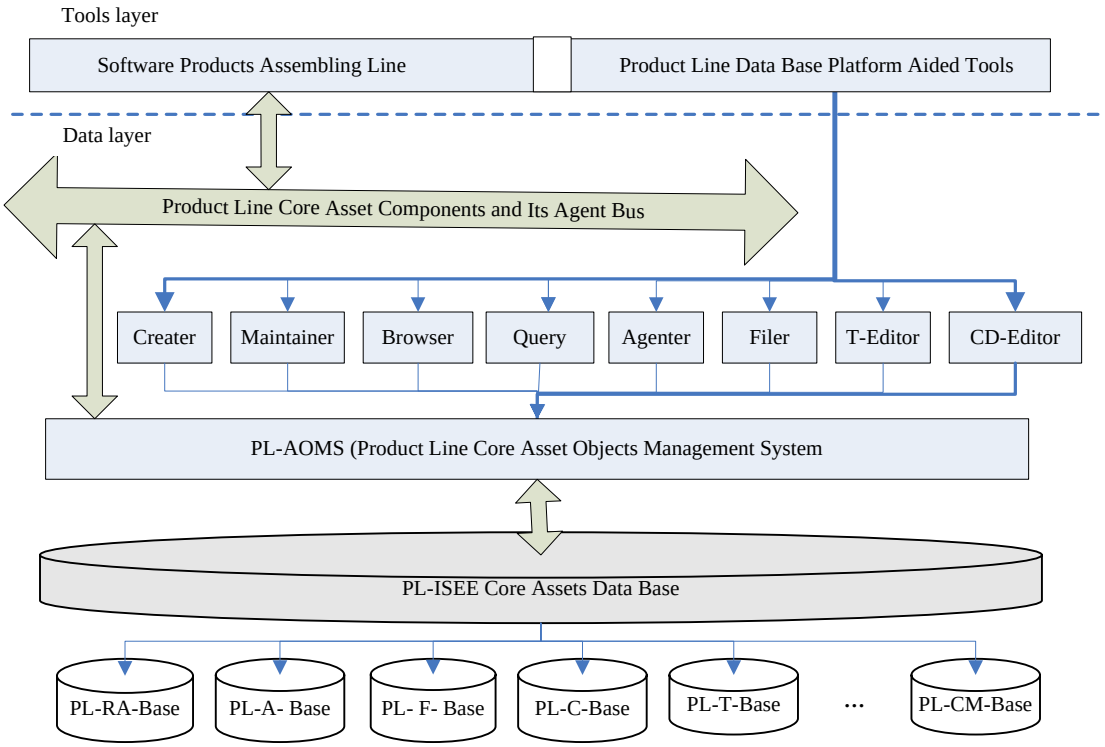

Figure 1. PL-ISEE Database Support Platform Architecture

line core asset databases are classified, associated, stored and managed according to the development process of production line or life cycle model separately, and all the asset bases will be used and transited smoothly during the production process of product line application software. For example, RA-Dbase is mainly used for storing the documents of problem domain analysis and project development plans in product line domain engineering and application engineering, and supported the analysis process of product line; A-Dbase, F-Dbase and C-Dbase, as the product line design asset databases are used to store design assets and support the design and implementation process of product line application software. TA-Dbase stores the test plans and test use cases of product line and supports the testing work of product line software development. CM-Dbase is used to support the version and configuration management of product line software.

It is must be emphasized that 1) the framework design of PL-ISEE database supporting platform must be based on organization structure and assembling mode of the product line core asset data. This kind of structure and mode addresses that architecture of one product line consists of several frameworks, one framework consists of several software components, and components contain computing components and connectors [13-15]. 2) There are two ideas in current product line developing process, one is the PLE (product line engineering) based on object oriented technology, it provides the services through the inheritance and interface of object classes for product line. Another one is PLE based on components, all application software products are assembled by using the third-party components in the PLE [16]. Thus, it is must be considered the classifying, storing and managing problems of source code or executive code asset components (include third-party components) in product line core assets database model and schema design. It is also the reason why we divide the component asset bases into source code component base and executive code component base. Without a doubt that this division bring a serious problem, that it is natural in description and expression of source code component objects by united OO model in database, but it is unnatural in description and expression of executive code component. Practically, the objects of OO theory and technology are any things and entities in real world. It is easy to achieve the object-oriented expression of executable code components by creating an executable class for the executable components which packages the executable component substance and relative features. Comparing with the executable component, the granularity of object oriented source code component class is much small, and this kind of component class is only reused in source code and depended on source programming language as well as not dispatched discretely. These defects restrict the application of source code components. Thus we hope the development and research of product line are based on third-party executable components, which brings new opportunity to the industrialization and automation of software production and the formation and development of new software engineering methodology. 3) The agent mechanism and distributed storage problems of the third-party components. The development and usage of the third-party components must through the register and release of agent organization on the Internet. This kind of agent

organization has been applied widespread in the Internet environment or integrated software developing environment. We can also consider proxy technology in product line core assets database system construction. This agent mechanism and distributed technology also provide convenience for collaborative development in different regions of product line.

IV. SCHEMA AND VIEWS DESIGN OF PL-ISEE DATABASE SYSTEM

A. The Schema Design of the Core Assets Database

A new industrialized PL-ISEE model with software assembly line ability is shown as figure 1 in reference [8]. The basic task of the data layer in the model is to provide all necessary components for software assembly line. So, How to organize product line core assets and provide convenient operations for searching and selecting asset components is the key to design and implement the environment database management system. Withal, the characteristics, classes, structures and association of product line assets and the organization mode and evolution mechanism of product line assembling process should be known during designing the architecture of PL-ISEE environment database system. According to the research state of the current product line, software domain engineering and application engineering, software architecture, software components and reuse technology as well as the features of product line industrialized production, the organization structure and model of product line assets is as shown in Figure 2 [14-15]. The asset elements assembling model of a software product line is composed by three different type asset elements which mean product line defined by software architecture, software architecture built by framework and framework composed by components.

requirement, business process, function and performance as well as environment restraint of specific application domain. A software product line which like the modern manufacturing industry assembly line only products a kind of software production as TV or cars. So far, the all-purpose product line is not yet existed in the real world. As the industry product line, a software product line should also be corresponded a specific application domain and implements the production of customized software cluster in the domain. Here, the product line architecture is just domain-specific software architecture and the blueprint of product line. The product line architecture is composed by a framework that can be reused and instantiated to a software part. In fact, the product line production process is really the process of assembling instantiated framework according to the product line architecture. Product line can be defined as a four-tuple:

Product-line architecture={Frameworks, F-Connecters, F-Constraints, F-Styles}

The product line is made up of the four tuple: frameworks, framework connectors, framework constraints and framework styles. Framework can be considered a class of large granularity reusable composite component, and framework connectors connect between frameworks, framework constraints describe the deploying conditions and operating rules of frameworks, and framework styles display the topology structure and computation mode among the frameworks.

The framework, as system-level and large granularity computing composite component, is defined as a five-tuple:

Framework={components, connecters, constrains, design patterns, hot spots}.

In order to effectively support the evolution process of product line framework, PL-ISEE core assets database

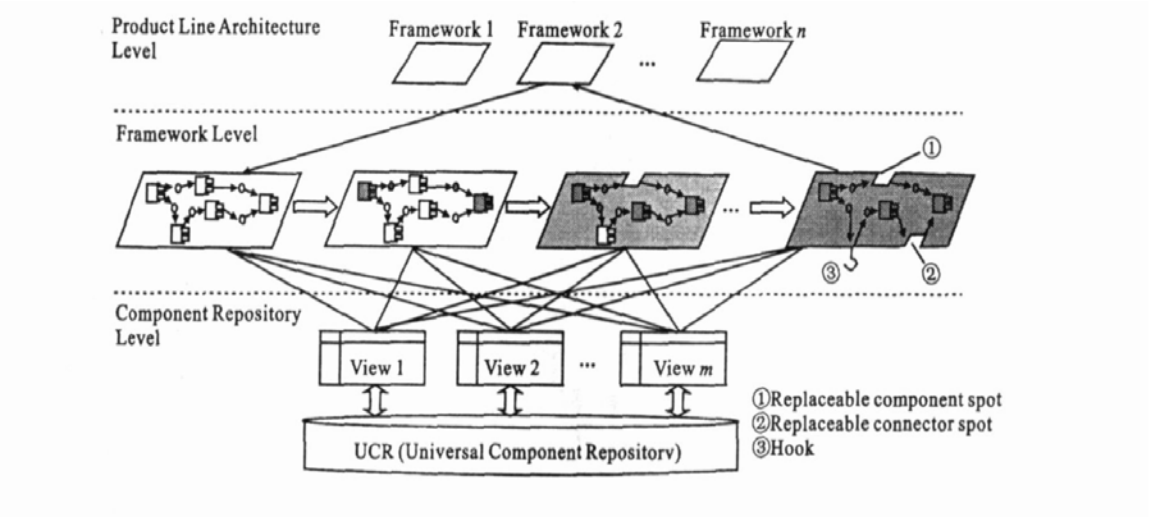

Figure 2. The organization structure and model of product line core asset components

In view of the relationship between the architecture of product line and special domain engineering, generally, DSSA (domain-specific software architecture) is built by the analysis models and specifications of the domain system will play an important role. This database stores all own or third-party reusable components information provided for potential developers to develop and assemble software products. The information includes

component body, interface, function, performance, constraints, domain information, deployment environment and so on. For recording and storing all component's organizational structure and associative relationships, components database must provide a kind of view mechanism. This view mechanism can reflect, express and record the mapping and associative relations as well as reuse grade of all the components generated the instantiate frameworks. The product line framework based on core asset component organization structure is shown in Figure 2. A product line can contain one or more frameworks, and every framework has a gradual changing process from the logical-level to the instance-level or blank-box. This gradual process will be to describe by the core asset components database schema and view.

B. The Views Design of the Core Assets Database

In order to support the instantiated evolution process of the product line framework, we offer a variety of views with mapping view, reuse view, the association view and coverage view in product line core assets database. Each view will describe and record the framework's changed process from the logical form to instantiated form, this process includes the assembling and evolution information from white-box framework to black-box framework. In order to clearly describe the instantiation process of product line framework, we reference [13-15] and give a framework definition as following:

Definition 1: Product line framework is the union of logical-level framework (the logical or abstract framework model) to be in product line problem space and the instance framework (the instantiated framework) to be in product line solution space. It means

$Framework = F_{logical\text{-}level} \cup F_{instance\text{-}level}$

Definition 2: Logical-level framework is abstract description and logical model of a group of components related specific domain business during product line early stage, and it is composed by meta-computing components, optional hooks, meta-connectors, topology and operation styles, constrains.

$F_{logical\text{-}level}$*={meta-components, hooks, meta-connectors, styles, constrains}*

Where, meta-components and meta-connectors are abstract or logical framework elements. They have their own interface, function, performance, constraints and so on information. In the framework instantiation process, they should be replaced by specific and eligible common components or connection (called as core asset components). Style is a definition of the topology structure and operation mode of framework components. Constraint is used to describe the domain constraints and operating conditions of framework components.

Definition 3: Instance-level framework is an instantiated assembling description of a group of common components related specific domain business during product line late stage, and this framework is composed by common components, replaceable component spots, hooks, common connectors, replaceable connector spots, styles and constrains. It means

$F_{instance\text{-}level}$ *={common components, replaceable component spots, hooks, common connectors, replaceable connector spots, styles, constrains}.*

The product line instance-level frameworks are obtained from instancing logical-level frameworks. While the framework is being instantiated, the logical-level white-box frameworks are evolving to instance-level black-box ones encapsulated common components and connectors. Meta-components are being replaced by common components and replaceable component spots, and meta-connectors are being replaced by common connectors and replaceable connector spots. Replaceable component spots and connector spots are separated from logical-level framework and interacted with the common components in the instance-level framework through the framework interfaces. Different from replaceable component spots, according to the demands of product line instantiation, hooks are optional.

According the definition of product line framework above, the process becoming form logical-level framework to instance-level framework is equal with the process replaced the meta-components with common components. At first, views mechanism must be provided to describe and save the mapping and associating relationships between the components of logical-level framework and instance-level one. In order to realize the assembly production of product line application software, some views and their implementing mechanisms are defined as follows:

1) Mapping-view: This view describes the corresponding relationship between meta-components of logical-level framework and specific common components in core assets database.

Let $M= \{c_1, c_2, c_3, \ldots, c_m\}$ is meta-components (including computing components and connectors) set in the logical-level framework. According to the historical records instantiated the framework, the set $C_1= \{c_{11}, c_{12}, c_{13}, \ldots, c_{1p}\}$ is a common components set which adapted to the meta-component c_1 and forms a one to many mapping between c_1 with $\{ c_{11}, c_{12}, c_{13}, \ldots, c_{1p} \}$. Similarly, the common components set adapted to the meta-component c_m is $C_m =\{c_{m1}, c_{m2}, c_{m3}, \ldots, c_{mr}\}$. Here, the set C_i $(i=1, 2, \ldots, m)$ is called the component item set. The set $C= C_1 \cup C_2 \cup C_3 \cup \ldots \cup C_m$ is called the complete component item set.

2) Using-rate view: This view reflects the use frequency or usage rate of component items set. High usage rate component items set have a higher reuse value and also easily forms common large-granularity composite component.

Let T is the specific components set of each instantiated framework. $T = \{t_1,t_2, t_3, \ldots, t_m \}$, where T contained in C, $t_1 \in C_1$, $t_2 \in C_2$, $t_3 \in C_3$, …, $t_m \in C_m$. D is the all items set of the instances of the framework, that is $D= \cup T_i$ $(i= 1,\ldots,n)$. A frequency of the emergence of a T component items set means the times (number) of the component items set T contained in D. Formula is:

$F(T_i)$= The number of T_i in D / Sum of the framework is instantiated in the all items set D

Naturally, while *F(Ti)* is larger, the reuse rate of component items set *Ti* is higher. Thus, *Ti* is the most stable and core common components set for developing practical framework products. So, *Ti* can be upgraded to a reusing large-granularity composite component.

Obviously, the use rates of core asset components are closely related to the life-cycle of production line and the time of core assets stored into the assets database. Average use rate of the core assets database is determined by the use frequency of each core asset component in the software products which have been developed. In stable and well-maintained product line, with the continuous development of new application software products, usage rate of the core assets will be gradually improved overall.

In addition to measuring the use frequency of component items set, the use frequency of the each component also could be measured. If *F(c)* denotes the utilization rate of component *c* in a period of time, $F(c)= n_c / N$, where n_c means the number of instantiated framework containing component *c*, *N* is the sum of instantiated frameworks in this period of time.

It may be confirmed that quantitative measurement of the use frequency of component items set and individual component is help for looking and collecting better reusing components or its set.

3) Association-view: This view describes the correlation and consistency between the components. In fact, the correlation of components is the probability of being used simultaneously in the process of developing the same product, and is defined as following:

If $C(c_1, c_2)$ shows the relevance and consistency of the asset c_1 and c_2, $C(c_1, c_2) = P(c_1/c_2)$ in which $P(c_1/c_2)$ is conditional probability of using both c_2 and c_1.

In the framework instantiation process, the selections of common components or components items set are gradually mature. According to the association views of asset components, developers can choose the next component or the next component items set based on the selected component or component items set.

4) Covering-rate view: Coverage rate is a quantitative measurement to describe the application scale and scope of the product line core assets database. It is expressed by the ratio of the number of existing core assets and the number of target core assets. Definition is:

Life cycle coverage: $C_L= n_l/N_L$, in which C_L is life cycle covering rate, n_L is the number of states covered in the life cycle, N_L is the number of stages planned phase-covered. Life cycle coverage is a quantitative description that the various stages (analysis, design, coding, testing, deployment, etc.) in the product line life cycle can be covered by elements in product line assets database.

Characteristics Coverage: $C_f = n_f / N_f$, in which C_f means the covering rate of product line characteristics, n_f is the number of features covered, N_f is the number of features planned coverage. Features mean the various functional features and non-functional ones determined by the domain needs, such as the clustering radio communications, visual production schedule and management, communication bandwidth, throughput, data security and etc. features in the port management product line.

Application Coverage: $C_a = n_a/N_a$, in which C_a means the coverage rate to some application fields, n_a is the fields which have been covered by core assets, N_a is the fields to be covered. The product line is mainly divided by the specific application field, such as logistics management, OA, CAD, and CAM and so on.

The above four kinds of views can guide developers to well determine the selecting strategy of the asset components in the process of instantiated frameworks. The mapping view can provide corresponding to optional asset component sets for each meta-component and meta-connector. The using rate view can provide the evidences of choosing the highest reuse rate and the most stable components. The association view can recommend next optional goals according to selected components set. The coverage views can be used to examine and determine the application fields and the using scale of the product lines.

V. THE CORBA IMPLEMENTATION OF THE PL-ISEE AND ITS DATABASE PLATFORM

According to the requirements of PL-ISEE model as shown in Figure 1 in reference [8] and CORBA architecture and its function configuration [17-18], we design an implementation framework of the PL-ISEE and its database platform architecture based on CORBA, and this framework model is shown in Figure 3. In Figure 3, in addition to retaining ORB functions of CORBA, that is, retaining ORB to be responsible for the communication and interoperability between customers and server objects in heterogeneous environments, the component model and component assembly capability provided by CORBA V3.0 cluster could be used. Furthermore, “core asset component and its agent bus” in the PL-ISEE model shown in Figure 1 in reference [8] can be implemented by ORB. In fact, ORB acts a dual role in the framework of PL-ISEE based on CORBA: one is ORB role in original CORBA architecture, another is product line core asset component and its agent bus which is also called CAB (Component Agent Bus).

The essence of PL-ISEE is to realize the assembly production of software products with product line core asset components (such as architecture, framework, compoment and connector). Therefore, in the product line development environment based on CORBA shown in Figure 3, the left part is client-side, which is work environment of all developers (such as analysis, design, programming, testing and management personnel) who produce the software products or core asset compoments. The right part is server-side, its main function is to provide enviroment database systen and implement the organization, saving, distribution, search, selecting and management of software product line core assets. Generally, prodution and management of product line

core assets will be completed by assets manufacturers or the third-party component providers. The communication, interoperation and compoment transmission between client and server side is achieved by object request broker (ORB) and ensure the transparency of these works and computing in heterogeneous environment.

Figure 3 shows that client-side environment framework mainly contains various types of development tools using software product line core asset components to assembly application software products, such as product line analysis, design, implementation and management tools. These tools would be integrated on the CORBA architecture according to the three-dimensional integrated model with interfaces, tools and data. It must be emphasized that the integration of client data (core asset components) appears at client side in the way of agent, which only is the registration and identification of components data. And actual components data is all located in the server side which will provide the retrieval, query, classification and management of component data with corresponding server object. The interface between ORB and client should be expressed by the interface definition language (IDL) provided by CORBA architecture, mainly including CS (Client Stubs), DII (Dynamic Invocation Interface), and IR (Interface Repository). CS is static interfaces provided by object server to client, that is, client local agent of server side object services, and customers call the services provided by service object through these interface. DII (Dynamic Invocation Interface) and IR (Interface Repository) can be used to dynamically call services provided by service objects under running status. On the server side (the right part of Figure 3), object adapter, product line object skeleton, DFI (Dynamic Framework Interface), component (include framework) object server and its component implementation database, architecture object server and its implementation database, management object server and management assets implementation database and other facilities are configured.

When the client users or developers use the services provided by component, architecture and management object server in server side, they could inquire and access the various services provided by service-side object implementation by two ways. One is that they could send the requests to object implement through client-side CS (Client Stubs) service agents of object implementation. Another is that they can use routines in IR (interface library) and send requests to service-side object implementation by calling DII (Dynamic Invocation Interface). Once object calling requests arrive at ORB, ORB is responsible for the requests and their parameters conversion, and sends the requests to the appropriate object adapter. After receiving the requests, the object adapter determines whether there is corresponding skeleton for the requested object implementation. If there is corresponding skeleton, object adapter call the operation in object implementation library through the skeleton. Otherwise, the object adapter calls the operations in object implementation library through the dynamic implementation routines in DFI (Dynamic framework interface). After finishing the service of object implementation, the results would be reversed back to client according to object request transmission and executing path. Here what must be noted is that object implementation or services of service-side mainly provide the search and acquisition of software product line core asset components and return to client-side developers through ORB in order to achieve the assembling production of product line application software [19-20].

In addition, the implementation of server-side product line core asset objects mainly complete the development, storage, classification and management tasks of the core asset objects (including product line architectures, frameworks, components and so on). These tasks may be

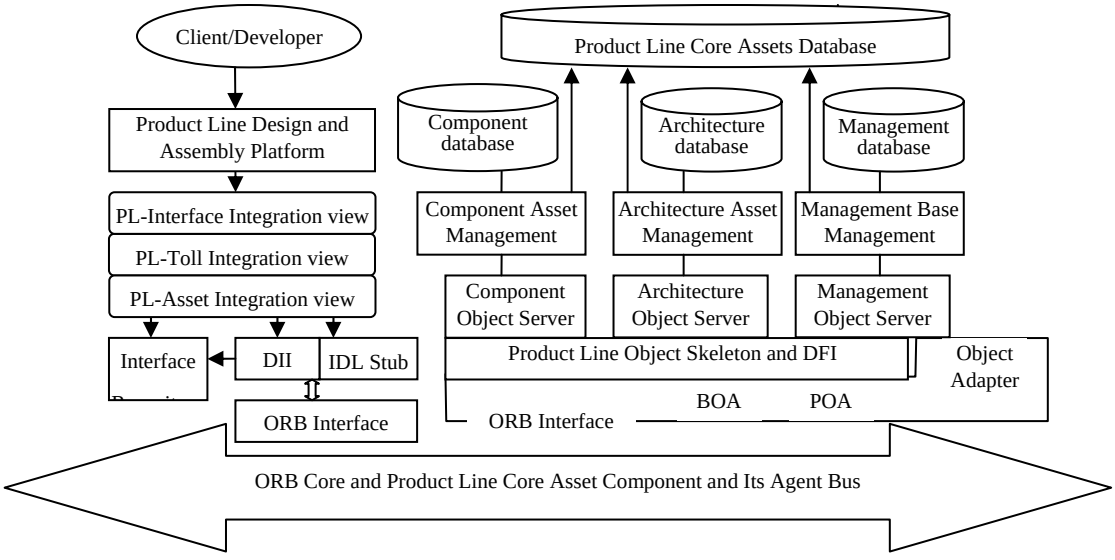

Figure 3. The Implementing Framework of PL-ISEE and Its Database Platform Based on CORBA

completed and provided by different manufacturers. Generally, the product line core asset objects mainly originate from three aspects: firstly, the extraction and reuse of the excellent components in existing system; secondly, new components proposed in the designing of new system; thirdly, suitable modifications and using of components in core assets database. In the management of core asset component objects, standardization and classification of various components directly affects the performance and application scope of whole component database system. The standardization of components requires the consistency of component interfaces and the unified messages format, pattern and protocol. The classification of components can adopt hiberarchy classifying methods(such as architecture, framework and compoment layer) to ensure the flexibility of the system and the security and scalability of the assets data transmission.

VI. CONCLUSION

In the paper, we systemically research on the architecture and realizing mechanism of a new industrialized PL-ISEE database supporting platform. According to the organization structure and assembling mode of the product line core asset components, we further discuss the framework, data model, database schema and views of PL-ISEE database supporting platform. This kind of database supporting platform can provide the various architectures, frameworks and components needed in the product line assembling producing process. Finally, the framework and realizing mechanism of the new PL-ISEE database support platform based on the CORBA technoloty are also systematically studied and discussed.

Assuredly, the above database platform framework and schema as well as views are very important to build ideal industrialized software product line core assets database system. Because this kind schema and views of database platform can search and select the asset components needed by assembly line from the core assets database and provide to the component bus, and then the bus as a conveying belt will transmit the components into assembly line.

But it is very complex and difficult to develop and implement the new industrialized PL-ISEE and its database supporting platform system, we still need to do a lot of works on product line engineering and database field. With the development and application of internetware, agent and cloud technologies, research on the new database supporting platform is more and more important and hard.

ACKNOWLEDGMENT

This project is supported by Fund of Jiangsu University Natural Science Basic Research Project, Grant No. 08KJD520013.

REFERENCES

[1] YANG FuQing. “Thinking on the Development of Software Engineering Technology,” *Ran Jian Xue BAO / JOURNAL OF SOFTWARE*, 2005, vol.16, no.1, pp.1-7.

[2] Zhang Xinyu, Zheng Li, Sun Cheng, “The research of the component-based software engineering,” *6th International Conference on Information Technology: New Generation*s, pp.1590-1591, 2009.

[3] Thurimella Anil-Kumar, Padmaja Maruthi T., “Software product line engineering: A review of recent patents,” *Recent Patents on Computer Science*, vol.3, no.2, pp.148-161, June 2010.

[4] Ahmed Faheem, Capretz Luiz Fernando, “An architecture process maturity model of software product line engineering,” *Innovations in Systems and Software Engineering*, vol.7, no.3, p p.191-207, September 2011.

[5] Unphon Hataichanok, Dittrich Yvonne, “Software architecture awareness in long-term software product evolution,” *Journal of Systems and Software*, v 83, no.11, pp.2211-2226, November 2010.

[6] Paol.Clements, Linda Nort-hrop. *Software Product Lines: Practices and Patterns* (SEI Series in Software Engineering). Addison Wesley/Pearson, 2003.

[7] Zhang YouSheng. *Software Architecture* (second edition). BeiJing: TSingHua University Press, 2006.

[8] Jianli DONG, Ningguo SHI. “A study on framework and realizing mechanism of ISEE based on product line,” *Journal of Software*, OCTOBER 2010, Vol.5, No.10, p.1077-1083.

[9] Dong Jianli. “Research on software engineering process model based on software product line architecture,” Computer Engineering and Design, 2008, vol.29, no.12, pp.3016-3018.

[10] S Chen, J. M. Drake, W.T.Tsai. “Database requirements for a software engineering environment: criteria and empirical evaluation,” Information and Software Technology, *1993,*Volume 35, Issue 3, pp.149-161.

[11] Lee Jaejoon, Muthig Dirk, Naab Matthias, “A feature-oriented approach for developing reusable product line assets of service-based systems,” Journal of Systems and Software, vol.83, no.7, July 2010, pp.1123-1136.

[12] De Amo Sandra, Mendona Manoel G., “Guest editorial: Special issue on databases and software engineering,” *Information Sciences*, vol.181, no.13, pp.2597-2599, July 2011.

[13] Wenhui HU, Wen ZHAO, Shikun ZHANG, Lifu WANG. “Study of Application Framework Meta-Model Based on Component Technology”. *Journal of Software*, 2004, Vo.15, No.1, pp.1-8.

[14] GUO Jun, ZHANG Bin, GAO Yan, GAO Ke-ning. “Research on Framework of Software Product Line Based on the View Mechanism of Component Repository,” *Journal of Chinese Computer Systems*, 2007, vol.28, no.2, pp.328-332.

[15] DING Jian-jie, LIU Yang. “Research on the Core Asset Library’s Evolution in Software Product Line,” *Journal of Shaanxi Institute of Education, 2009*, 25(1):81-83.

[16] Len Bass, Paul Clements, Rick Kazman. Software architecture in practice (Second Edit ion). *USA: Pearson Education, Inc*, 2003.

[17] YAN Juan, ZHANG Lifang, CAI Xuqing, “Research and Development of CORBA,” *Software Guide*, Jun, 2009, Vol.8, No.6, pp.41-43.

[18] CHEN Bo, LI Zhoujun, CHEN Huowang, “Research on Component Model s :A Survey,” *COMPU TER EN GINEERING & SCIENCE*, Vol.30, No.1, 2008, pp.105-109

[19] LI Yuling, ZHANG Yongmin, "Integrated Environment for the Software Enterprise Based on CORBA," *Journal of Henan Agricultural University*. Vol.39, No.1, Mar, 2005, pp.116-120.
[20] HE Tianzhang, WANG Panqing, LI Xiaohui, "Research and Application on CORBA Component Assembly," *Science Technology and Engineering*, Vol.8, No.1, Jan.2008, pp.84-86.

Jianli DONG was born in Shanxi province, China, in 1957. He got his Bachelor of Mathematics Science in Northwest Normal University, Lanzhou, Gansu province, China, in 1988 and got his Master of Software Engineering in Beijing University of Aeronautics and Astronautics, Beijing, China, in 1995. He is now a professor at the School of Computer Engineering in HuaiHai Institute Technology, Lianyungang, China. He has published 70 papers, and completed over 8 scientific research projects, and won 6 times scientific and technological progress awards from the province and military.

Mr. DONG current research interests include software engineering, integrated software engineering environment, software architecture, engineering database system, object-oriented technology.