Assiduousness of Domain production in Secretarial Executive System

Author(s): Mukesh Kumar[1], Dr. Praveen Kumar[2]

[1] DCSE, [2] Professor, DCSE, NIMS University, Jaipur

ABSTRACT

In the foregoing dissertation, I tried to put forward language description formalism called Collages. It can be used to engineer Domain Specific Languages (DSLs), which are computer languages made to solve problems of specific domains. Here the focus on DSLs which have algorithmic design and are supposed to be used in corporate environments. Domain specific language can be broken in three parts, these are abstract syntax, description language conceptualization & relationship among them and last is their constraints which encode rules of domain. Domain's concrete syntax produces graphical and textual presentation of abstract syntax elements. Their semantics meaning are normally defined operationally. Operational semantics normally encoded system behavior and could be described as a collection of "elements", each denoting the transformation This paper present on sphere feature model, sphere architecture design and area implementation in an enterprise. This paper demonstrates the accounting management feature modeling based on the extended (Feature-Oriented Domain Analysis) FODA method and system architecture of accounting management domain, integrates Aspect Object Oriented Programming technology with domain implementation, and designs a whippersnapper AOP framework based on the object proxy pattern to separates crosscutting concerns in the domain implementation phrase. Research result shows this method can effectively seal insulate and abstract variability in requirements of accounting management domain, instruct the designing and implementation of accounting management components, get the requirement of software reuse, resource sharing and collaboration in accounting management domain.

Keywords

Feature-Oriented field Analysis, facet Object leaning instruction, whippersnapper.

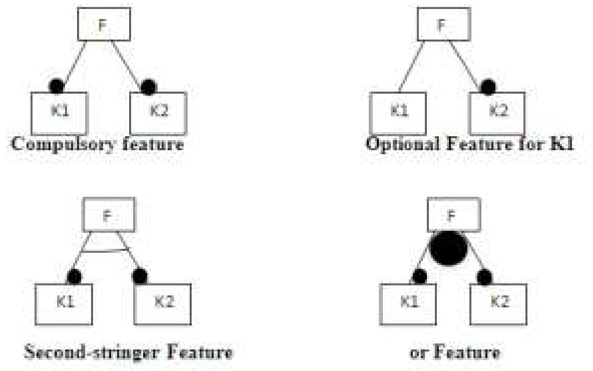
1. INTRODUCTION

The focus and contribution of this dissertation is to design and elaboration of a language engineering discipline, based on widely spread state-based intuition of algorithms and programming. It opens the possibility to apply DSL technology in typical corporate environments, where the beneficial properties are of smaller, and therefore by nature, it is more secure and much more focused on computer languages. This dissertation does not cover the equally important topic as to how formalize these beneficial properties by means of declarative formalisms and how to apply mechanized reasoning and formal software engineering to DSLs. Domain engineering is a reusable approach that focus on a selected application domain as like inventory control, finance management, word processing etc. The motto of domain engineering is find, catalog, construct and broadcast set of software artifacts that could apply for future software in specialized application domain. In domain engineering, we perform domain analysis and capture domain knowledge in the form of reusable software assets. By reusing the domain assets, an organization will be able to deliver a new product in the domain in a shorter time and at a lower cost. In industry, domain engineering forms a basis for software product line practices. Domain engineering is most often divided into three phases: domain analysis, domain design, and domain implementation. At present, from the point of domain engineering, little research has been carried on the accounting management domain. Based on the real project, this paper introduces domain engineering method into the development of accounting management system. In the domain analysis phrase, we use the FODA method to analyze the accounting management domain, expand its feature- oriented modeling method, establish the feature model of accounting management domain; in the domain design phrase, we design multi-tier system architecture of accounting management domain; In the domain implementation phrase, We combine AOP technology with OOP technology, separate crosscutting multi modules concerns in software, reduce the dependence between components effectively. Practice has

proved the systems developed by this method have a better performance of maintainability, extendibility and reusability.

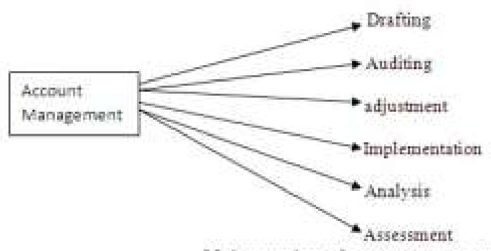
1.1 Attributes Oriented Domain analysis

In this text it is not focused on the problem of how to describe the syntax of a language, but in practical applications of DSL design, the definition of syntax is the first, and thus most critical task. Many successful DSL applications show very simple, sometimes line based syntax styles. Another approach for avoiding syntax problems is to use XML for the representation of programs. A method specifically designed for DA is the Feature Oriented Domain Analysis (FODA) method developed at the SEI. This process is for domain analysis which supports the discovery, analysis, and documentation of commonality and differences within a domain. The feature oriented concept emphasis on findings the capabilities that are normally expected in applications in a given domain. The FODA domain model captures the similarities and differences among domain assets in terms of a set of related features. A feature is a distinctive aspect, quality, or characteristic of the domain asset. The features identified by the FODA method can be used to parameterize the system product line and Implementations of the domain assets. The features differentiating domain entities arise from differences in capabilities, operating environments, domain technology, implementation techniques, etc., i.e., a range of possible implementations within the domain. A specific implementation consists of a consistent set of feature values describing its capabilities. The feature diagram depicts the decomposition of features into sub-features in a hierarchical way. For each sub-feature below a certain feature it can be specified if it is compulsory, second-stringer or optional. The graphical notations introduced in are used here. We first briefly describe the representations used in illustrated in Figure 1. The compulsory feature is represented by being attached to an edge ending with a filled circle. So the feature F consists of both K1 and K2 in this case, and the feature instances here are {F, K1, K2}. The optional feature is represented by being attached to an edge ending with an unfilled circle. So the feature F may or may not contain K1. The optional feature instances here are {F, K2} and {F, K1, K2}. The second-stringer feature is represented by connecting edges with an arch. So the feature F consists of exactly one of its child features. The second-stringer feature instances here are {F, K1} and {F, K2}. Note that if K1 is optional while K2 is compulsory, then the second-stringer feature instances here are {F}, {F, K1} and {F, K2}, because the child feature instances derived from the K1 side contain an empty feature. The OR feature is represented by connecting edges with a filled arch. The OR feature instances here are {F, K1}, {F, K2} and {F, K1, K2}. If there is an optional child feature, then the OR representation is actually equivalent to the situation that all the child features are optional, i.e., the OR feature instances will be {F}, {F, K1}, {F, K2} and {F, K1, K2}.



1.2 Stain model of account control area

This choice is on one hand restricting the application of the current implementation to real-live programming languages with simplified syntax only, but on the other hand it simplified both the implementation of the tool, and the specification work with the tool. If we would have chosen a full fledged solution with completely independent treatment of concrete and abstract syntax, as featured by most of the mentioned attribute grammar and formal semantics systems, it would not be possible to design, implement, test, and validate a new programming language prototyping environment from scratch. Through domain analysis, we find common and variant features of different account management systems, from different requirements: business requirement, user requirement, and functional requirement. Business requirement depicts business ability that the software system should have. User requirement depicts the interaction process between user and system, and this process may reflect the generally accepted business process in this domain. Functional requirement depicts functions that software system must have in order to realize the specific business requirements. Through domain analysis, we divide the service of account management domain into the following types: Account Drafting, Account Auditing, Account Implementation, Account Adjustment, Account Analysis, Account assessment. Among them, account assessment is optional features.



Major services of account management domain

The second analysis is to identify functional features which the service has, analyze the specific functions which systems must have in order to complete special service. Taking account implementation control service as example, its functional layer includes compulsory features and optional features. And as shown in Figure 2, Compulsory features include execution account drafting, execution account auditing, execution account management and query analysis. Optional features include data import. Compulsory features, namely common features, exist in each member system of the special domain, but optional features are one type of representation style of variant features, and only exist in parts of member system of the special domain. Optional features represents the variability which is relative to whole features, its introduction enables the feature model to respond the different system's diversity of domain, and makes the feature model to have better tailorability and expansibility.

The third Behavior characteristics layer analysis. The task of behavior characteristics layer analysis is to identify behavior characteristics what the function should be there, analyze behavior features of the early stages of functional implementation, such as preconditions of functional implementation, preparatory works; analyze the principal behavior characteristics of function part, find its outstanding features and its possible variability; analyze behavior features of the later period of function implementation, such as the postposition condition of functional implementation and the domination shift after the functional implementation.

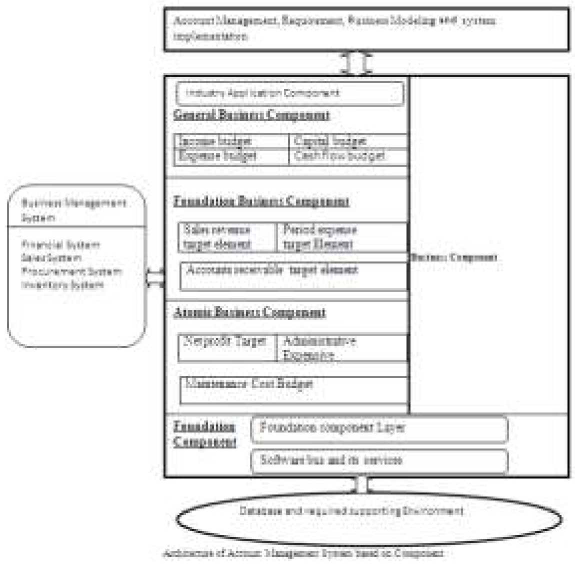
Description executive Domain with Architectural Design:-

Domain designing is the core architecture for a family of applications according to domain analysis model, namely a Domain-Specific Software Architecture (DSSA), and based on the DSSA, We can identify, develop and organize the reusable components. According to the requirements defined in the domain analysis stage, considering the actual development environment (such as operating system, database, communication mechanism, middleware, and so on, this paper designs Account Management domain architecture, This architecture uses the hierarchical architecture style. The hierarchical architecture style can avoid system component’s coupling, protect and divide system function, improve maintainability, reusability and extendibility of software.

This domain architecture has five components: foundation component layer, atomic business component layer, foundation business component layer, general business component layer, industry application component layer.

(1) Industry application component:- This component is designed to satisfy special industry business requirements. It can be encapsulated by one or more atomic business components, or by one or more foundation business components, and even also can be combined by atomic business components, foundation business components and general business components.

(2) General business component: - This component is a subsystem level application component which is formed by encapsulates foundation business components or atomic business components, such as revenue budget components, investment budget components, capital budget components, cash flow budget components.


Account Management, Requirement, Business Modeling and system implementation
Industry Application Component
General Business Component
Income budget
Capital budget
Expense budget
Cash flow budget
Business Management System
Financial System
Sales System
Procurement System
Inventory System
Foundation Business Component
Sales revenue target element
Period expense target Element
Accounts receivable target element
Business Component
Atomic Business Component
Netprofit Target
Administrative Expensive
Maintenance Cost Budget
Foundation Component
Foundation component Layer
Software bus and its services
Database and required supporting Environment

Architecture of Account Management System based on Component

(3) Foundation business component: - On the basis of atomic business component, these components are able to complete certain business functions through aggregation of some atomic components. This type of component faces to application directly, such as sales revenue target components, period expense target components, business interface components.

(4) Atomic business component: - According to the decomposition business object, this is made by encapsulation of various types of foundation components. This level usually includes the following component types: representation components (forms according to object's method) data components (forms according to object's attribute).

(5) Foundation component: - This component is the lowest level in this architecture, and it is the core support to implement the business object function. It takes Database, Document, Mathematical formula, Documentary evidence and so on as the object, carries on the code level encapsulation according to component standard, forms general representation components, data components, operational components or generic component template. The components of previous layer may call it directly.

Accomplishment of resources Management area

In the part of domain design, we have putted required and harder structural DSSA and assigned the stable parts to the budget management domain system architecture and the variable parts to components. In the process of component implementation, we normally use OOP Object-Oriented Programming) for the simplifying the things and encapsulating the class. Aspect-oriented Programming (AOP) is a new programming technology which compensates the weakness of Object-Oriented Programming (OOP) for applying common behavior that spans multiple related object models. AOP introduces Aspect, it packages the behavior which impacts multiple classes into a reusable model, it allows programmers to model crosscutting concerns and eliminates the code tangling and scattering caused by OOP, the code is more readable and easier to maintain. The key to achieve AOP is to intercept normal method call. In order to complete some extra requirements, we will need to add extra features transparent "weaving" to these methods. Generally speaking, the weaving method includes two major types: Static weaving method and Dynamic weaving method. Static weaving method usually need to extend compiler's function, directly weave codes into the appropriate weaving point by modifying byte codes(Java) or IL code(.Net). Or, we need to add new syntax structure for original language to support AOP. As for dynamic weaving method, there are many specific implementation methods. In the Java platform, we can use Proxy pattern, or custom Class Loader to implement AOP features. Generally, at the .Net platform, the following methods can be used to achieve the dynamic weaving method:

(1) Use Context Attribute and Context Bound Object to intercept the object methods.

(2) Use Emit technology in the run-time to build new class which codes are woven into.

(3) Use Proxy pattern

2. CONCLUSION

In this paper it is depicts the application of domain engineering in account management system development. Domain analysis method of FODA this paper has extended its feature oriented modeling method and design multi-layer framework according to the domain analysis result. At the domain implementation segment we applied a lightweight AOP framework with the name of SJAOP. This technology with the help of OOP separates crosscutting multi modules concerns in software, reduces the dependence between components effectively, and implements the system with a better performance of maintainability, extendibility and reusability.

3. REFERENCES

[1] James A. Hess, William E. Novak, A. Spencer Peterson,"Feature-Oriented Domain Analysis (FODA) Feasibility Study", Carnegie Mellon Software Engineering Institute (SEI), USA,2013.

[2] Steven Kelly and Juha-Pekka Tolvanen. Domain-Specific Modeling : Enabling Full Code Generation. Wiley-Interscience, 2014.

[3] Wang Qian-xiang,Wu Qiong,Li Ke-qin,Yang Fu-qing,"An Object-Oriented Method for Domain Engineering",Journal of Software, vol.13, no.10, pp.1977-1984, 2012.

[4] Li Ke-qin, Chen Zhao-liang,Mei Hong,Yang Fu-qing,"An outline of Domain Engineering", Computer Science, vol.26,no.5,pp.21-25, 2013

[5] Wang Fan,Tan Guo-zhen,Wang Hao,He Qin-lai,"Feature Modeling Method Oriented to Traffic Domain Component", Computer Engineering, vol.35,no.1,pp.280-282, 2009.

[6] Zhengmin Liu, "Research on Enterprise Budget Management System Based On Domain Engineering". International Journal of Digital Content Technology and its Applications, Vol. 5, No. 9, pp. 88 ~ 94, 2011

[7] Kai Chen, Janos Sztipanovits, and Sandeep Neema. Towards a semantic anchoring infrastructure for domain-specific modeling languages. In International Conference On Embedded Software, pages 35–43, 2010.