

Designing reliable software using DAF

Rakesh Agarwal, Bhaskar Ghosh and Ajit Sarangi

Infosys Technologies Ltd., India

{rakesh_a}{bhaskarghosh}{ajitsarangi}@inf.com

1 Introduction

Reliable software demand has been increasing for the past two decades but still there exists a gap between the demands placed on the software industry and what the state-of art currently can deliver. There has been lot of work in mechanics of program construction but there has been little progress in improving the practice of software development. Competitive pressures are causing companies to show how their software developed is superior to others. The justifications that companies can provide are based on the market information, market characterization and domain characterization.

Software development has been considered as a variant on the paradigm of an expenditure of intellectual labor[1][2]. The growth of software systems has been drastic over the years. The most promising approach to the reducing of the cost of software development is to reduce the amount of software to be developed. This is possible if we use proper software process models in the development of the software that contribute to the development of some generic stuff that can be reused.

The institutional process by which reuse can “possibly” be adopted in an organization is derived from the model first proposed by Prieto-Diaz[3]. The importance of this model is its identification of two distinct, sequential activities of *assessment* followed by *implementation*. This two-step approach allows reuse adoption to be systematic in nature rather than opportunistic.

The development and implementation of new software systems in a company is not an easy task. Many projects in this area have failed and have cost a lot. To deliver successful projects it is necessary that we have a structured developmental process and a repository that can act as reference model [4]. In order to cope up with the industries requirement we have studied various developmental projects and come out with a sound framework, called, Design Architectural Framework (DAF). This will assist in development of application software in most effective manner in terms of quality, time and money. Further, this will help to build a reference resource library that can be referred for further development projects in the same domain. The paper introduces the DAF for developing complex, reliable and robust software. It identifies the deliverables that are produced in any software process, which can be reused..

2 Generating reusable components

Software reuse[1][5] aims in the development new applications from existing software and is a general principle that is instrumental in avoiding duplication and capturing commonality in inherently similar tasks. It offers the potential to streamline and simplify software development, greatly reduce the time, cost, and effort needed to develop and

maintain high quality software and increases productivity. The reusable components can be classified in four broad categories: Business domain level, Functional level, Design level, and Lower level

The business domain level will capture all information that is domain specific. Functional model will gather the functionality of the domain in the form of Data Flow diagrams (DFD) and system flow charts. The design model will capture the Entity-Relationship diagrams (ERD), State-Transition diagrams, etc. The lower level will capture the reusable program structure. Reuse can only be successful in the context of a domain [6]. The reuse of assets that are specific to a particular domain is called vertical reuse, while the reuse of generalized assets is called horizontal reuse.

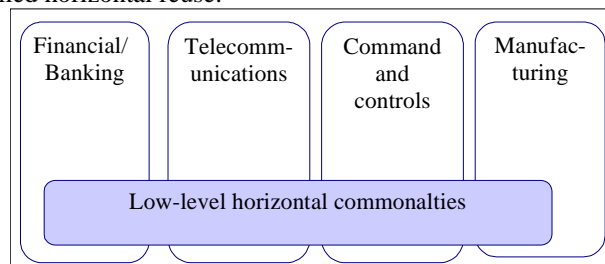


Figure 1: The vertical/horizontal scope of reuse

The vertical-domain-oriented reuse of software asset is possible only if there are some means to deliver generic components to the reuse library. If the generic architecture is available then the reusability is possible via interconnection mechanism.

In order to establish knowledge reuse we propose a model given by Capilla[7]. Domain and Reuse Engineering stores in the repository the extracted knowledge through domain analysis processes. Domain analysis is defined as “the analysis of relevant information of the problems to generate a model of information or knowledge that can be reused to build applications”. It is composed of two components that are development **for** reuse (Domain Engineering) and development **with** reuse (Reuse engineering). **Figure 2** shows the process of domain and reuse engineering.

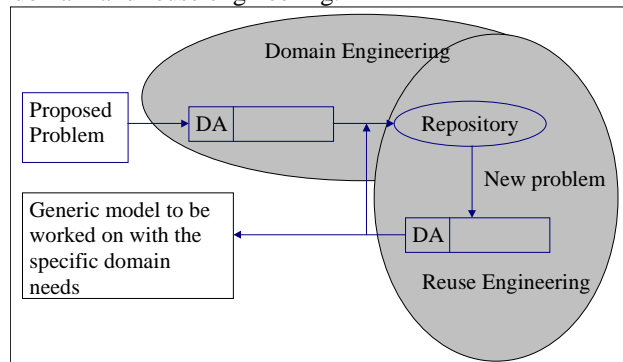


Figure 2: Process of domain analysis and reuse engineering

