The K-model is being proposed as a solution to the Software reusability & reengineering practices during component-based software development. The development of new software is mainly depends on assembly of existing software components, software reengineering along with new development. The main development phases may overlap with each other. The main emphasis of K-Model is on reusability during software development and development of reusable software components which may be useful in upcoming projects of same kind. Reusability means the use of composition techniques during software development; and it is achieved by selecting reusable components and assembling them, or by reengineering the existing software to a point where it is possible to pick out components for development. Frame working attempts to identify related components and establish relationships among them within the same application domain. Assembly implies collection of reusable components from same application domain. Reusability within K- Model is more efficient and cost effective than traditional models since it integrates domain engineering, frame working, reengineering and the storage & retrieval mechanisms in reusable component repository to achieve it.

K- Model provides a time and cost saving formula, which enhances our capabilities to focus on specific aspects of software application rather than general aspects of application. Therefore, the development of a component should be done in such a way that the less importance is given to satisfying the specific needs and more focused should be given on reuse and reengineering of an application that is being developed. The specific parts of a design are those parts which change a general set of components into a specific software system for a particular application.

The K-Model supports development with reuse and reengineering through component collection, and their cataloging.
The components are selected, adapted and reused with help of composition techniques. In existing software’s, the same method can be implemented by using the concept of reengineering. At the end of software development, there may be many new reusable components those needs to be verified, catalogued, classified and further stored into reusable libraries.

4.1 Phases of SDLC in K-Model

- Domain Engineering
- Frame working
- Reusable Software Components repository
- System Requirement Analysis
- Partial Design
- Assembly of reusable components from repository
- System Design
- Implementation & Integration
- Testing
- Reengineering & Archiving
- Maintenance

Figure 4.1: K-Model
### 4.1.1 Domain Engineering

The process of analyzing an application domain in order to determine areas of commonality is called Domain engineering. Thus, domain engineering is an action that should be implemented at the beginning of software specification if reuse is to be considered. Software requirements, provided functionality, objectives and constraints of the system are very much of interest throughout the system analysis and domain engineering phases. Thus, the understanding of real-world application is very important and an abstract model of that application should be illustrated. Therefore, the boundary between system analysis and domain engineering seems fuzzy because identifying key abstractions in the application domain may be viewed as part of system analysis or domain engineering. Domain engineering is also concerned with the identification of potentially reusable components. Domain engineering is the activity of collecting, organizing, storing and reusing past experience in building software systems or parts of software systems in a particular domain in the form of reusable assets.

In K-Model, software systems can be categorized as per application or business areas like airline reservation systems, medical record systems, portfolio management systems, order processing systems, inventory management systems, etc. In same way we can also categorize components of system software as per their functionalities like database systems, synchronization packages, workflow systems, GUI libraries, numerical code libraries, etc. We refer to areas organized around classes of systems or parts of systems as domains. Characteristics and requirement sharing are there by the components or system within the same domain. Thus an organization can take advantage from number of systems or components of domain when developing subsequent components in same domain. New products can be delivered in short time and on low cost by capturing acquired domain knowledge in form of reusable assets. Domain engineering is a systematic approach to achieve this goal and it has three process components, these are domain analysis, design and implementation.
In the development of new software system, domain engineering is complete process of reusing domain knowledge. It’s a main concept of systematic system reuse. Application domain is key idea in systematic software reuse that contains system sharing commonalities. Most of Organizations are limited to small range of domains and repeat same systems with small changes to meet customer needs. These organizations reuse portions of the previous systems to develop new system rather building a system from the scratch. Domain Analysis is process of identifying domains bound them and find out their common and variable features. The information collected is captured in models and used to identify the artifacts during implementation phase such as reusable components, domain specific language or application generators that can be further used to develop new systems.

The aim of domain engineering is to get better quality in developed software products through reusability of artifacts and it shows that most of new systems are variants of the old one in same field. Domain engineering results in growth of the business profits and reduces time of development with the use of previous system’s concepts and applying these to achieve the target system. The reduction in cost is evident even during the implementation phase. From the previous experiences it shows that code size is reduced by 50% in both i.e. number of methods and number of symbols; reduction in line of code is nearly 75% with the use of domain specific language. The main focus of domain engineering is on capturing the gathered knowledge during the software engineering process.

The reusable artifacts make it possible to reuse the components at minimum cost and better quality in new software systems. Domain engineering, like application engineering consists of three primary phases: analysis, design, and implementation. Unlike software engineering which focuses on a single system, domain engineering focuses on a family of systems. A good domain model consists of a repository of knowledge about the domain definition, characteristics and specifications.
**Domain analysis:** In K-Model, we used domain analysis technique to define domain, collection of information and results in a domain model. Domain analysis aims to identify the common points in a domain and the varying points in the domain. It is possible to develop configurable requirements and architectures by the use of domain analysis rather than static configurations which are the results of a traditional engineering approach. Requirements derived from traditional approaches are ineffective to produce configurable requirements which make domain analysis significantly different from requirement engineering.

It is required to apply reuse in the earlier phases of SDLC so that domain engineering can be effectively applied. Derivation of domain analysis is based on the artifacts results from the past experiences in the domain. Potential input sources of domain analysis are existing systems, their artifacts (such as design documents, requirement documents, user manuals), standards, and customers. Domain analysis does not solely consist of collection and formalization of information, a creative component exists as well, which makes it different from requirement engineering.

Aim of engineers during domain analysis process is to extend knowledge of the domain beyond already stored information and to classify the domain in commonalities and differences. Within the range of domain it mainly produces a domain model consisting common and varying properties of system by producing architectures and components in a configurable manner. Domain model helps to create a foundation upon which these components can be designed. An impelling domain model not only consists of varying and consistent features but also defines the concepts, ideas and phenomena, and vocabulary used within the system. Feature models decompose concepts into their required and optional features to produce a fully formalized set of configurable requirements.

**Domain design:** Domain design takes the domain model that results from domain analysis used by K-Model and target to develop a generic architecture system which adapts all the systems. Similarly application
engineering uses functional and non-functional needs of system to develop a design called domain design which takes configurable requirements produced during the analysis phase and give a configurable and standardized system for family of systems. Domain design target to develop architectural patterns that have commonalities and solve same problem within the domain rather than having dissimilar patterns. Aim of design model is to preserve the flexibility extended by the developed model and to satisfy as many domain requirements as possible. The architecture should be sufficiently flexible to meet all systems within the domain and stiff enough to provide strong framework to base the solutions.

**Domain implementation:** Domain implementation is the creation of a process and tools for efficiently generating a customized program in the domain.

### 4.1.2 Frame working

A framework can be seen as a generic structure that gives a skeleton for developing software in a definite application domain. Frame working builds an interrelationship perceived important within the application domain and seeks to identify components. Such identification of components may arise from the well-known functionality common to that application domain. Considering the example of application domain of railway reservation system where various entities of system can be seats, trains and passengers; and interrelationships can be; reserve a seat, select a particular train and so on. So, the entities having important relationship among these entities can be structured into a framework according to their semantic meaning in that application domain. During frame working it is required that developer might have sketchy idea about candidate components for reuse. On the other hand, as framework consists of sets of components that convey a design for a family of related applications, sometimes modification in developing software to fit it in available framework is beneficiary to increase the productivity.
It is easy and fast to create and customize the software rather than starting it from the scratch, sometimes frameworks are not generally useful outside their domain as they have domain dependent components. The proposed life cycle model emphasis on reuse of software components already developed and classifies the components to form new frameworks. Rather than focusing on individual application, here the aim is to develop a workbench that includes software components and generic application framework that features the software in a particular application domain.

A software framework provides reusable platform used to develop applications, products and solution. A software framework consist of support programs, compilers, code libraries, tool sets and application programming interfaces that collect all the components to enable production of a project or a solution. Frameworks have main differentiating features that separate them from normal libraries:

- The overall program’s flow of control is not directed by the caller like in libraries or normal applications but by the framework.
- A framework has a default behavior. This default behavior must absolutely be useful behavior.
- A framework can be extended by the user normally by selective overriding or specialized by user code to provide specific functionality.
- The framework code is not allowed to be modified, excluding extensibility. Users can extend the framework, but not modify its code.
- The overall development time can be decreased if the designers of software architect intend to initiate development by allowing designers and developers to dedicate their time for meeting software necessities rather than working on more standard i.e. low-level details which is not much required, of presenting a working system. For example, a team using a web application framework to develop a
banking web-site can target on writing code particular to banking rather than the mechanism of request handling and state management.

- A phenomenon termed "code bloat" is usually added to the size of programs by frameworks. Competing and complementary framework results in a product due to customer demand driven application needs. It is required to give additional time for learning the framework. This criticism is clearly valid when special or new framework is faced by development staff. If such a framework is not used in consequent job tasking, the time devoted in learning the framework can cost more than purpose-written code familiar to the project’s staff.

- Future projects are easy and faster after understanding of framework; this concept of framework is to make one-size-fits all solution set with familiarities, code development should logically rise. No allegations can be made about the size of the code after all assembled with the output product, nor its relative efficiency and conciseness. The use of any library solution automatically drags extra and unused assets unless the software is a compiler-object linker making a tight (small, wholly controlled, and specified) executable module.

### 4.1.3 Reusable Software Components Repository

The main aim of software components storage and retrieval system in K-Model is to construct the repository in which components can be stored and searched on the basis of static and dynamic behavior of a software component. A software component’s entity, description and metadata are stored in repository. The three can be stored together or discretely. The discrete scheme is chosen to reduce burden, better openness, convenient for expansion and maintenance. A component repository is categorized into expressive and an entity repository.
The software component retrieval system depends on metadata, ontology faceted categorization and the model consists of three layers i.e. view, application and data layer. The view layer is a web form that provides searching interface for various components’ users and library; admin interfaces for administrators and experts. Whereas the application layer responsible for expressing components, their categorization, administration, feedback, authority, logs and realize the view layer’s result. Data Layer consists of three databases i.e. describing repository, component repository and Meta data repository. The metadata repository contains information about special domains, give accurate query terms, and reduce some phenomena such as different names and same names with separate meanings. Describing repository can give some important information such as interfaces, functions, administrative levels, applied domains, developed languages, applied environments and editions of software components. The component repository store information about components and gives services such as download and more information on the semantics and also derives knowledge for interrelated software components. The gathered domain information should be related to components via an analysis of their aim and functionalities and makes it possible to relate software components association with in the domain semantics.

Retrieval of components is based on the analysis of semantic information. A user normally input query terms using interface of the software component retrieval which is then matched with the repository terms. Best fit expressive terms are selected for feedback but if terms didn’t match accurately then terms are selected from repository using heuristic algorithm which are further filtered and refined so that accurate query terms are formed. Describing repository reflects the accurate needs of user depending on the faceted categorization by a module of accurate query processing ,appropriate software component will be searched by a fix algorithm; user filter appropriate software component repository of component. Then search results can be improved and optimize by
applying neural network algorithm. Hence the role of (re) user to find the best-fit component from the search result would be much easier.

**Search Optimization using Neural Network:** The main aim of this tool, used into K-Model is to categorize reusable software component using Feed Forward Back Propagation Neural Network (FNN) and Levenberg-Marquardt (LM) as a training algorithm. LM algorithm used in this study makes training process quick as the solution is approached. For this study, sigmoid, hyperbolic tangent functions are applied in the learning process. Attribute and characteristics of reusable component software are used by FNN to classify the reusable component software. FNN also classified reusable software component in type1, type2, and type3. FNN is created by generalizing the gradient descent with momentum weight and bias learning rule to multiple layer networks and nonlinear differentiable transfer functions. FNN is trained by input vectors and corresponding target vectors. Neural network trains until it classify the defined pattern. The training algorithms use the gradient of the performance function to determine how to adjust the weights to minimize performance. The gradient is determined using a technique called back propagation, which involves performing computations backwards through the network. The back propagation computation is derived using the chain rule of calculus. The input vector is composed of 28 elements, corresponding characteristics and attribute values of reusable software component. One hidden layer is determined empirically to be 16 and the output layer consists of 4 neurons. In addition, the transfer functions of both i.e. hidden and output layer is tan-sigmoid. For the training of neural network, the target is four element vectors.

### 4.1.4 System Requirement Analysis

Initial requirements are collected from customer, analyzed and then compared with the requirement document of similar project taken from repository. Then requirements are refined with customer involvement. A document that gives complete description of the behavior of system to be
developed and may consist of set of use cases that express interactions the users with system is known as System Requirement Specification (SRS). It also contains non-functional requirements. Non functional requirements apply constraints on the design or implementation. SRS document lists all the important requirements that are needed for project development. A clear understanding of product is required to derive the requirements. This is prepared after detailed communication between the project team and customer.

Initially a project begins when a business need is identified and a manager is appointed to manage everything in project. These business needs are documented in concept proposal and after getting its approval the system concept phase begins. The scope of system is identified by system boundaries and then it requires approval from senior officials for funding before the planning phase. The concept is further developed to express how the system will influence the employee and customer privacy after implementation of the approved system. Various things like project resources, activities, schedules, tools, and reviews are defined to make sure that product and services provide required capability on time and within budget. Functional requirements are defined and trace the requirement in terms of data, performance, security and maintenance of the system. In K-Model, all requirements are defined to a level, sufficient for systems design to proceed. All requirements need to be measurable and testable and relate to the business need or opportunity identified in the initiation phase.

### 4.1.5 Partial Design

In K-Model, the initial requirement collected are designed with help of UML diagrams and then compared with the stored UML diagrams in the repository for searching software components of similar projects.
4.1.6 Assembly of Reusable Component from Repository

In Assembly, various reusable components from specific domains are focused. Different mechanisms are there to achieve reusability when different kinds of reusable components are involved. Composition reuses the most basic components and it can be seen as a process of developing a piece of software from elementary self-contained components; although object-oriented development accomplishes reusability through inheritance by reusing classes.

4.1.7 System Design

During this phase physical characteristics of system are designed and a detailed design of system using design documents of similar system is prepared. The operating environment is developed; inputs and outputs of major subsystems are defined and resources are allocated to processes. K-Model’s guidelines instruct users to reviews every documented input or approval required by the user. The physical characteristics of the system are specified and a detailed design is prepared. Subsystems identified during design are used to create a detailed structure of the system. Partitions of subsystems into one or more design units or module are done and a detailed logic specification is prepared for each software module.

4.1.8 Implementation & Integration

Implementation of the new generated design and integrating it with assembly of the existing code components those are to be reused in current project. Here the coding of designs is done; computer programs are written using some programming language or program generator. Programming tools like compilers, debuggers and interpreters are used for coding. With respect to the type of application, the right programming language is chosen. The result document developed during the design phase is converted into hardware, communication and executable software. In this phase, K-Model provides a systematic approach to test, integrate and retest the software. Hardware is assembled and tested. All
the modifications are made in system under a production environment, where system modifications are installed and made operational. The phase is initiated after testing and acceptance of system by user and will operate according to defined user requirements.

4.1.9 Testing

During this phase system is well tested and reengineering practices are applied, if needed, else the system is archived in repository. Individual modules are coded and tested separately. Then all the modules are tested as a complete system to ensure that interfaces between modules work correctly with expected volume of data on intended platform and work as per the user requirements. Various functional requirements are also tested to ensure the satisfaction of customer after integration of various components. System also needs to be certified and accredited before deployment in production environment.

4.1.10 Reengineering & Archiving

Combination of different processes such as reverse engineering, re-documentation, restructuring, translation, and forward engineering is known as Re-engineering which is a study, analysis and alteration of an existing software system. In K-Model our main aim is to get the existing software and then re-implement it to improve its functionality, performance and implementation. New techniques are used to add new functionality. Reengineering concept is more time saving and cost effective in comparison to design a software system from scratch; it extends the functionality of the existing system.

**Reverse Engineering and Forward Engineering:** A higher level of abstraction is achieved by identification of components and their interrelationship in the process of reverse engineering. Everything like requirements, design and architecture is recaptured in reverse engineering. Technical details, interactions, information and rules about application and process are also captured. Aim of reverse engineering is
recovery of lost information and achievement of higher abstraction and facilities for reuse.

New system is produced by moving downward from level of abstraction, and replacing existing system information with detailed and modified information. This downward movement from a sequence from design to implement is called forward engineering.

**Archiving:** Reusability not only consists of existing components in new software but also to develop components meant for reuse. Some of the components can be generalized for reuse in future during development of system software. An important consideration in the quest of reusability into K-Model is how to make a potentially reusable component available to future projects.

Archiving should reflect the activities involving cataloging and storage of components. Components are required to be well written, and well documented so that can be easily understood. Additionally, extensive cross-referencing is necessary. Some of components differ in complexity, scope and levels of functionality. This dissimilarity in components makes it tough to create single database of software assets. This argument in component technology results in observation; several interconnected reusable components are more effective than a single universal library of components. So, we decide to develop a centralized repository of components to develop some of specific application than to create a single library.

**4.1.11 Maintenance**

Software after installation undergoes changes and needs to be maintained from time to time with new requests. Changes can happen due to many reasons such as unwanted input values in system that can directly affect the operation of system. The software should be developed to accommodate changes that could happen during the post implementation period.
During operation system is monitored and modified according to user requirements. In K-Model, process review is used to assess system from time to time and determines how it can be made better and effectively adapted to meet an organization's needs. When modifications or changes are identified as necessary, the system may reenter the planning phase.

4.2 Work Flow in K-Model

In present era of software development, general projects are not dependent on a specific domain; instead knowledge of more than one domain is required for software development. On the other hand, almost every project is designed & developed with a practice of using more than one SDLC model.

Initially the different application domains are needed to study in order to find similarity between various applications within same domain. Then a skeleton is prepared, and same can be used for developing similar applications in that domain. A skeleton basically is a structure of multiple components those can be integrated/fitted into either existing component from the repository or newly designed.

Initial requirements are collected from customer, critically analyzed and refined. Then certain UML diagrams are designed based on initial requirements. These UML diagrams are given as input to our tool which stores these UML diagrams in MDL file format and search the repository to find related components. These related components may be of any type. We receive the output in descending order of weighted components. These weights are assigned according to the percentage of matched contents.

Now this result of searching tool is given as input to our NN based optimization tool for further optimization of search results. The output of the tool is then manually analyzed.

After analyzing the above output, the related requirement documents are critically studied and gold plating of initial requirements are performed with due interaction with the customer.
In the next step, these refined requirements are converted into system design; further, this new design is implemented by developing some new components and integrates these new coded components with reusable components retrieved from the repository. Final code is then tested against defined testing criteria, if there is any correction required then it is to be reengineered otherwise it is stored into reusable components repository as per archiving concept. The tested software is then deployed and maintained as per requirements.

### 4.3 Strengths of K-Model

- Increased productivity and efficiency by using domain engineering practices
- Cost effective and timeliness due to reusability of existing components.
- Most suitable for similar kind of projects those are developed within same domain
- More user involvement during requirement phase
- Iterative support
- Risk reduction