3. MATERIAL AND METHODS

This third chapter “Material and Methods” describes research material and methodology. Research methodology is a procedure used to logically develop an understanding of the research topic. Scientific methodology is accepted as a part of research methodology as it is useful in creating new knowledge and to solve the residing problems. Scientific methodology is a structured enquiry that is generally applicable. In the procedure of scientific research, classification of data, systematic observation, and interpretation of data have been done. This chapter also helps to present wide-ranging logic to select methods, cases and tools used in this research. Thus this chapter will cover the research process.
3. Material and Methods

The Rich Client Platform provides a generalized model to assemble an application from components, which are provided by different software vendors. These vendors can build components by implementing the plug-in contract, whose primary purpose is to specify, which requirements a plug-in must fulfill to extend another plug-in, and wrapping their software into plug-ins. This chapter will describe how the Eclipse plug-in model works, how single plug-ins can interact and how existing software patterns contributed to the development of this mechanism. To describe the plug-in model in this chapter, the Intro plug-in, which is provided with a standard Eclipse M08 installation, will be used. Later on, Earth Navigator plug-ins will be introduced and explained in detail.

3.1 PLUG-IN CONCEPT

To provide certain functionality, a plug-in is a computer program that interacts with another program. Typical examples are plug-ins for media players, like video or audio codec, or for graphical applications, like filters. The main program defines the interface and the plug-in contributes by implementing this interface.

Generally spoken, there are two main differences in plug-in concepts: the form of plug-in execution (direct or indirect) and the plug-in identity. On the one hand, the direct approach is to provide enough information about existing plug-ins and by this way other plug-ins can execute this plug-ins directly.

This alternative reduces the overall amount of interfaces in an application, because a single direct execution replaces two execution calls that would be needed in an indirect plug-in execution architecture, where a service broker is responsible for executing plug-ins. The direct plug-in execution reduces the amount of data that has to be saved in the service broker and improves the performance of the application. The indirect plug-in execution architecture provides a service broker that does error checking when failures occur during the plug-in execution. So the service broker promotes reliability. In addition, it hides changes to the application programming
3. Material and Methods

interface (API) to make plug-ins more portable. For the reason that there is a single point of contact, this architecture even supports the modification of the application.

Plug-in discovery and registration is also a major concern of a plug-in architecture. A service broker can discover the identity of plug-ins instead of providing an interface where plug-ins can register their identities when they are added or removed. Self-registering of the plug-ins improves the performance during the service execution because the service broker knows about existing plug-ins. The disadvantage is that it requires the plug-in developer to add code to register itself and if the platform application programmings interface changes, he will have to modify the code. Looking up plug-ins at runtime is less error-prone and thus makes the application more modifiable and reliable. Furthermore, the code required to look up plug-ins does not increase by adding new plug-ins.

With the introduction of OSGi, Eclipse supports the direct plug-in execution on top of a service registry. Therefore plug-ins do not have to register them but are discovered during start-up. From this point on they can be invoked by other plug-ins.

3.1.1 Mechanism

It is very clear in the figure, that after registering the plug-ins on the host application it can provide services. Plug-ins do not usually work by them. Host application is responsible for the services which are provided by the plug-ins. End-users allowed to add and update plug-ins dynamically; this is made possible by making the plug-ins independent of host application. By this way it is possible make changes to the host application without any interference.

Sometimes third parties need some plug-ins to interact with the host application to provide a standard interface to open application programming interfaces. Third-party plug-ins can continuously work in proper way if there is a change in the original version if it is using a stable application programming interface. The life-cycle of obsolete applications can be extended here.
3. Material and Methods

3.1.2 Traditional Plug-Ins versus Pure Plug-Ins

In the traditional plug-ins, it uses the downloadable software bundles so that it can provide the enhanced functionality of the hosting applications. This functionality is achieved by linked to extension mechanisms with well-defined interfaces although they are not compiled into the particular application. Mostly it is not necessary to access to the source code of the application at the time of building the plug-in. Only those functions can be implemented via plug-in, which are known to the host application. These functions are made activated when needed. Most people are familiar with that there is everything is a plug-in if we talk about new pure plug-in architectures.

At the time of composing a larger system which is not restructuring, it is necessary that architecture should be the way that it supports the extensibility in regard of plug-ins by plug-ins. If there is no host application then some plug-ins provide the functionality of hosts, by providing the guidance to other plug-ins in well-defined popper points that at which points other plug-in
can provide the additional functionality. The points which are talking about are the extension points. Whenever a plug-in implements the extension point, means it adds an extension. To explain the extensions which are provided by the plug-ins to client plug-ins, there are extension model defined. Extension models also describes that in which ways, they can be extended [31].

3.1.3 The Eclipse Plug-in Model
A Plug-in can be seen component that extends the platform with certain services and functionalities. It can seamlessly be integrated into the platform at deployment time. At runtime, the component is wrapped into a plug-in runtime class. This class is responsible for the management and configuration of the plug-in. The main purpose of this class is to perform some special tasks during the activation or deactivation of the plug-in.

At deployment time, all required resources of a plug-in must be copied to the plugins directory of the Rich Client application. These resources consist of the manifest file, the jar file with the assembled class files and other resources like images, documentation, etc. The plug-in is activated when other plug-ins request one of its functionalities. A user interaction usually results in the instantiation and initialization of such a plug-in.

“Our knowledge for solving software engineering problems in the right manner is more and more being put in a nutshell in tools which provide a good interface to do so. These tools provide the greatest of benefits when they operate in surroundings that can provide complete integration with existing elements such as editors, compilers, debuggers, profilers and visualizes. A major challenge is to develop some tools which can enhance the way of developing projects, which can help in changing the future environments” [32].

3.1.4 The Manifest File
The Manifest file documents the XML elements and attributes that define plug-ins. The manifest file provides the information about the unique id org.eclipse.ui.intro of this plug-in, its provider name, its name its version number, and its main class org.eclipse.ui.intro.internal.IntroPlugin.
3. Material and Methods

The “%” signs in the name and provider-name attribute are a reference to a properties file that is located in the same folder. The runtime element defines the name of the target library as well as the resources that should be exported to this library.

The very small Eclipse core has the task to load and execute the plug-ins, which are located in the plugins folder. The Eclipse core itself is a plug-in and contains the library runtime.jar and a plugin.xml file. The runtime.jar library belongs together with the plug-in org.eclipse.platform with the Java archive startup.jar and the plug-in org.eclipse.core.boot with the Java archive boot.jar to the absolute minimum requirement of an Eclipse based application. Moreover, boot.jar is responsible to finish the installation process of a newly installed platform. Another basic requirement is the plug-in org.apache.xerces with its Xerces-DOM implementation. The DOM is required to be intelligent to interpret the manifest files [33].

3.1.5 Dependency and Extension

As far as Eclipse model is concerned, a plug-in may have the relation with another plug-in. This relation can be of two types:

1. Extension. There are two main types of plug-ins in this relationship. These are extender plug-in and host plug-in. The functionality of a host plug-in is extended by the extender plug-in.

2. Dependency. There are also two main types of plug-ins in this relationship. These are prerequisite plug-in and dependent plug-in. The functionality of a dependent plug-in is supported by prerequisite plug-in.

All these kind of relationships are declaratively specified at manifest files of the plug-in with using the XML extension and elements which are required for this purpose. In addition a reachable plug-in may as well stay deactivated when its functionality is not used during the lifetime of the running instance.
3. Material and Methods

3.1.5.1 **Dependency**

For the reason that dependency statements are compile and runtime directives, the statements in source reference 3.2 lead to the loading of prerequisite plug-ins at runtime and to the augmentation of the class path at compile time.

3.1.5.2 **Extension**

To make the functionalities of a plug-in available to the user, a plug-in has to be intelligent to contribute menu items, toolbar items, etc. to the user interface. Due to the specification of Eclipse, this has to be possible without recompiling the application or altering the user interface classes, in this case org.eclipse.ui, where the menu is setup. So the org.eclipse.ui plug-in has to be augmented. As it does not know anything about the plug-ins which requires it to be altered, it has to define a set of configuration and behavioral requirements. This set can be thought of slots, also called extension-points in the Eclipse terminology. So in this way one plug-in can define a couple of extension points, for example to allow its menu and its toolbar items to be altered. Furthermore, one extension-point allows one or more extension plug-ins to be plugged into it. These extension points allow extending the host plug-in by adding processing elements. In the role of the host plug-in, the Intro plug-in, offers an extension point to add Introduction related configurations to its presentation layer and the Intro plug-in, in the role of the extender plug-in, adds a standard configuration and implements the actions which should be executed when invoking this configuration. Also other plug-ins extends the configExtension extension-point of the Intro plug-in, like for example the JDT. In this case, the extension is purely descriptive. It tells the Intro plug-in the text, which it should display when showing this item and the link that it should follow when clicking on it. A single extension-point can be extended one or more times by a single extension (e.g.: various tutorial items of the JDT) and one or more times by a couple of extensions for example JDT, PDE, etc. Callback objects, through which the extender and host plug-ins communicate, are not always required because the extension model allows providing the
necessary classes, which are needed by the host plug-in to change its behavior, at compile-time. The extension can be purely descriptive.

As already mentioned, not all extensions lead to a callback in the extender plug-in. Some extensions may be used to set flags in the host plug-in or to describe a certain behavior. The configExtension of the JDT, for example, only tells the host plug-in about the URL that it should follow when the user clicks on a certain link. So the extension does not provide a callback object, but it causes an object, in this case the link on the introduction page, to come into existence.

The creation of new top-level workbench menu items, to the configExtension extension-point, the action Sets allows at the point the extension. For the reason that processing a click on a top-level menu only resides in exposing the lower-level menu entries, this action does not need further processing from the extender plug-in and therefore no custom callback object. When designing an application that is based on the plug-in architecture of Eclipse, one achieves great flexibility when it comes to exposing parts of the application. The application can be parameterized through XML tags and therefore be extended with complex extension structures.

3.2. PLUG-IN DEPLOYMENT AND ACTIVATION

During the deploying a plug-in in an Eclipse installation, there is need of copying some resources like jar files, the manifest file and some other resources. These things are copied into one folder of the plug-in, this folder is saves in the directory of the installation’s plugin. When it is required to perform some special functionality, the required plug-in then can be activated by the Eclipse runtime. Activation of a plug-in consists of loading of the runtime class and after loading; the instance of this class is initialized.

When plug-in is activated and deactivated, some special processing is required. This work is done by plug-in class. This also includes the allocation and reallocation of the resources also. If simple plug-ins is concerned, there is no need for activation and deactivation. For example, for the JUnit plug-in, there is no need for activation and deactivation. In this case the plug-in
designer does not provide any specific plug-in class. If this happens, a default plug-in class is provided by the Eclipse runtime for the plug-in instance.

![Diagram of plug-in design](image)

**Figure 3.2 Plug-in**

The plug-in designer subclasses org.eclipse.core.runtime is used to activate or deactivate the specific plug-in. For startup the plug-in provides the methods of the foundation means these methods are overridden and for shutdown there are deactivation methods. It provides the fully-qualified name for the plug-in subclass which is the value of the corresponding plug-in manifest file’s attribute class.
Eclipse *platform is a* plug-in management kernel of Eclipse framework. This plug-in is present in every Eclipse deployment. This is core firm plug-ins. The identities are hard-coded in core plug-ins and necessarily activated in each running instance of Eclipse. If we talk about Non-core plug-ins, it is only activated by other plug-ins when required.

As far as Eclipse model is concerned, a plug-in may have the relation with another plug-in. This relation can be of two types:

3. **Extension.** There are two main types of plug-ins in this relationship. These are extender plug-in and host plug-in. The functionality of a host plug-in is extended by the extender plug-in.

4. **Dependency.** There are also two main types of plug-ins in this relationship. These are prerequisite plug-in and dependent plug-in. The functionality of a dependent plug-in is supported by prerequisite plug-in.

A non-core plug-in can be activated in a running occurrence, has been deployed in an Eclipse installation. But for that it should have transitive relation with a core Eclipse plug-in by the union of the extension associations and the dependency. When there is need for the extendibility the functions of another plug-in and its functions are required to bear to, a plug-in will be activated.

If any deployed plug-in which is unreachable from any core plug-in is there then the possibility is that may not deployed from the point of view of plug-in activation. And if there is any inactivate plug-in of a running instance is unreachable then no user triggering event or other action elicits its use.

### 3.3 DEPENDENCY

If there is dependency of a plug-in on some another plug-in for some functionality then in the plug-in manifest file of that plug-in the dependency of that plug-in should be specified via requires element. In the plug-in manifest file compile-time directive Dependency and runtime Dependency both are defined. At compile time, for compiling a dependent plug-in, Eclipse can
be directed to augment the classpath by all of its prerequisite plug-ins via jar files. When the dependent plug-in is activated a prerequisite plug-in can be made available to that particular plug-in. Eclipse is responsible for the same at runtime.

3.4. EXTENSION

One or more user interface elements to be needed to add to the base Eclipse workbench. This is done when the services of a plug-in are needed to make available to the user directly. For example help menu items added to the workbench user interface to make the workbench's help plug-in available to all the users.

As discussed above, an extension is a method of adding some element or some processing elements to a plug-in. However, this procedure is not necessary to do with user interface elements. By adding processing elements to a plug-in may permit other plug-ins to extend it. For the same, an extender plug-in indefinite extension and this causes to change a host plug-in’s behavior. This behavior changes means- including some add-ons to processing elements to the host plug-in, and some customization to the behavior. This includes additional elements to be provided by services provided by the extender plug-in up to some extent. All the way through which the corresponding of host and extender plug-ins is possible, Extension is done. In some simple cases, a single callback object is added to the environment like single act of extension. If we talk about the callback object, it is similar neither to the host plug-in nor to the extender plug-in objects. And as these objects are dissimilar, a callback object is used here. A callback object having the aim of instantiation is managed particularly by provider-supplied code of Eclipse platform. This callback object is a plain old Java objects. In a single extension, it can add more than one callback object. There is an example of it in Eclipse platform that, it is allowed to add the set of menus to its user interface, and it using only one single extension.

Custom callback objects are provided by an extender plug-in provides which are not fundamentally required. An extension declaration serves merely to parameterize instances of built-in classes. These classes are recognized to the host plug-in at compile-time with the
purpose of the kind of behavior modification. It is possible that required of a host plug-in can be provided totally by objects.

It is not necessary that the host plug-in directly exposes aspects of each of its extensions in its interface as the extension model does not need it. For example, an extender plug-in be notified, an extension may simply ask that of certain events known to occur in the host plug-in independently of the extender plug-in, without doing any changes in the host plug-in's interface of that extender.

The workbench user interface allows both its editors and its menus to be extended. Different kinds of extensions are permitted by a plug-in for the improvement. But it is necessary that every time, the extension must match to a unique set of behavioral and arrangement requirements. An extension-point is obviously a point which allows any number of extensions to be plugged into it. Therefore for providing this kind of functionality, different types of slots is provided by an extensible plug-in so that extensions can plug into. These types of slots are called extension points.

*Extension-point* and *Extension* are standard Eclipse plug-in terms. *Callback object, extender plug-in, and Host plug-in* are some terms which are used for the different roles of objects in an extension.

3.5. THE OBSERVER PATTERN VERSUS THE ECLIPSE EXTENSION MODEL

As it is noticeable, the observer pattern is similar pattern except that the listener extension has its static deployment-time registration of listeners. So the observer pattern may be thought of as a specialization of the Eclipse extension model if we ignore the dynamic nature of observer registrations. In fact, the extension model of Eclipse known as modulo dynamic registration, adds control to the observer pattern. This has been done at a number of levels:

1. **Fixed Notification Semantics in Subject versus Arbitrary Semantics in Host**
Based on the configuration of the extension, a variety of customizable responsibilities can be accepted by the host, under the extension contract, for example the provision of user interface elements. An extension instance can be associated with arbitrary parameterized host semantics. There is no such customizability and parameterization in the observer design pattern.

2. **Uniform Treatment of Observers versus Differentiation of Extenders**

All observers are treated by the observer pattern in a given subject uniformly, notifying every one of an observable event. In contrast, Depending on the specifics of the extension-point contract and the parameters of an extension, different extensions of a given extension-point can also be treated differently.

3. **Callback Bundle versus Single Callback**

A single observer provides a single callback object whereas a single extension may provide multiple callback objects.

If Eclipse extension model and the extension model of a microkernel are compared then internal servers, for example, OS device drivers plays an essential role in it. A core set of services are allowed by both models which is to be extended by additional provider-supplied services. The internal extension model of the microkernel architecture is generalized by the Eclipse extension model in two ways.

1. A packaging mechanism for sets of related extensions is provided by the Eclipse plug-ins.

2. In Eclipse extension-points may be provided by any plug-in. This feature makes it extensible, so that it can be extended by other plug-ins.

In compare of it, in the microkernel architecture, the microkernel core has exclusive standing which can be seen as the sole extensible component in a system.
3.6 Principle Facilities of This Abstract Model

The principle use of this architecture is the Eclipse workbench. The basic extension model is an abstract architectural pattern which is quite different from its specific materialization in the workbench. A powerful and general paradigm for architecting extensible systems is provided by the plug-in extension model of Eclipse based on components which are loosely-coupled.

The principle facilities of this abstract model are:

1. **Extension-points**-

   A plug-in defines an extension-point that stands in a *host* role with respect to the extension-point, and this may be extended by one or more plug-ins that stands in an *extender* role with respect the extension-point. An extension-point is embodied in particular way in which a plug-in allows it to be extended. There is an agreement related with each extension-point. The agreement puts responsibilities on both the extender plug-ins and the host.

2. **Deployment-time pluggable components**-

   A plug-in can be viewed as an instance of a plug-in class in a running system. At the time of deployment Plug-ins are assembled into a system at manifest file, of each plug-in are specified declaratively. These characteristics are interpreted at runtime to instantiate the plug-in and can be related to other plug-ins with the help of it.

3. **Obligations of the Host**-

   On the basic of behavior of the host, additional requirements are included in the host obligations under an extension-point contract. For example, additional processing elements augment a requirement on the host to its interface.

4. **Obligations of the Extender**-
Manifest file have the extender declaratively describes the characteristics of an extension. An XML schema is needed by the extension-point contract for this description. The extender's manifest file must confirm to this schema through the extension specification. Slots for the concrete classes of the extension's callback objects are included by the schema, and for the parameters required to construct these objects. The concrete classes must confirm to expect interfaces defined by the host and are furnished by the extender. The host instantiates the configured callback objects at runtime based on their configuration parameters.

5. **Extensions as Parameterized Callback Bundles**

An extension-point requires extenders to provide custom implementations known as callback objects for these interfaces and contract generally provides one or more callback interfaces. Then the host is forced to call back on these callback objects under some conditions which are specified in the contract and based on a particular extension's configuration parameters.

3.7 **THE ARCHITECTURE OF RICH CLIENT**

An RCP application can be seen as three layers-

1. **Presentation layer**

2. **Business layer**

3. **Data layer**

1. **Presentation layer** - The presentation layer more often holds presentation logic components and beside this, it has User Interface.

2. **Business layer** - The business layer holds business entity components with the dealing workflow. It also consists of business logic.
3. **Data layer**- The data layer holds basically service manager components. It also facilitates the user with data access.

Rich client applications mainly have a presentation layer. The main work of this presentation layer is to contact a remote business layer which is hosted on server machines from beginning to end services. For example, an application which is used for data entry can send all of the data to the server where it is processed and stored.

Some applications may be difficult that only communicates with other services and stores the data to consume or send back information. These applications perform most of the processing themselves. For example spreadsheet software like Microsoft Excel that stores both the data and state locally, beside this it performs complex local tasks itself, and only communicates with remote servers at the time when it needs to fetch and update linked data. The guidelines for the data and business layers in such applications do not different from other applications; it is just like general applications. These types of rich clients can have their own data access layers and business layers.

### 3.8 Design Considerations for Development with Rich Client Application

At the time of designing of rich client application, the main objective of the software designer is to design the formation in such a way so that it could reduce complication. This can be done easily by separating tasks into different areas of concern and by choosing an appropriate technology for his application. As these both works have a great impact on the application, it should be done carefully. The design should fulfill all the requirements for the application in terms of security, performance, ease of maintenance and reusability. These all things are essential for a robust application.

The following guiding principle should keep in mind when designing rich client applications-

- **Appropriate technology should be chosen on the basis of application requirements**-
Suitable technologies for designing the rich client application are XAML, Browser Applications, Windows Forms, WPF, and OBA etc.

- **Presentation logic should be separated from interface implementation**-

  The use of separate components like presentation logic, interface implementation within the application makes the application robust as this can reduce dependencies, promote reusability, and make maintenance and testing easier. That’s why some design patterns like Supervising Presenter and Presentation Model that separate user interface logic from user interface rendering provides ease of maintenance, improves testability and promotes reusability.

- **Presentation flows and presentation tasks should be identified**-

  This will help us to design each step and each screen with the help of multi screen or Wizard process.

- **Designing should be in the manner so that it could provide a usable and suitable interface**-

  The features of application such as navigation, layout, localization and choice of controls should have maximum usability and accessibility.

- **There should be separate concerns for all layers**-

  There should be separate components for each work like components for data accessing code should be separately located in a data layer, extract business rules and other tasks which are not related to presentation should be located into a separate business layer.

- **Presentation logic should be in a way that it can be reused**-
There are a lot of components that can be reused in other application such as helper classes, generalized client-side validation functions and libraries that contain templates can be reusable in several applications.

- **Round trips should be avoided when accessing remote layers**-
  Whenever remote layers are accessed, they freeze or block the user interface so coarse grained methods should be used or executed asynchronously if possible.

- **Client should be loosely couple from any remote services it uses**-
  For communicating with services which are located on separate physical tiers, a message-based interface should be used.

- **Tight coupling should be avoided for the objects which are located in other layers**-
  Abstraction should be used which is provided by abstract base classes, or messaging when communicating with other layers of the application or common interface definitions.

### 3.9 SOME IMPORTANT DESIGN ISSUES

There are quite a few common issues which must be considered at the time of development of design of the application. These issues can be divided into further categories into specific areas of the design. The following points help us in resolving the common issues in each area-

- Business Layer
- Communication
- Composition
- Configuration Management
3. Material and Methods

3.9.1 Business Layer

In a typical thick rich client, the business layer is located on the client itself. A typical thin rich client acts as the interface to a business system. And the business layer is one of the parts of that business system and is typically open to the elements as a service. However, the following guidelines should be considered when designing the business layer for a rich client:

- Identify the service interfaces and business layers that the application is going to use. If these both things are identified correctly then this can help to minimize coupling between the client and services that it uses or we can say a remote business layer. In this case the application will import the interface definitions, contact remote services to access them, and write the needed code that accesses these service functions by using the interfaces.

- There can be two things about business logic that it may contain sensitive information or not. If our business logic does contain sensitive information, then business layer should be placed on a separate tier. And in another case if our business logic does not contain any sensitive information, then some of the business rules should be placed on the client side. This will definitely improve the performance of the User Interface as well as performance of client application.
3. Material and Methods

- When the remote server starts up then the developer have to decide the business rule information which is obtained by client from the remote server. This is how the client will obtain information which is required to operate business rules as well as other client-side processing. Beside this, this information should be automatically updated whenever there are some changes. Some important aspects should be considered like how users or administrators will update the business rules whenever requirements get changed.

3.9.2 Communication

If a rich client uses other remote services or if the business and data layers of a rich client application are situated on a remote tier and open to the elements as services, then in this case, by using a number of methods and protocols, it can communicate with these services. These may include Simple SMTP e-mail messages, HTTP requests, SOAP Web service messages, remote database access protocols, DCOM for remote components, or custom communication protocols or other TCP/IP-based standard. If both the data layer and business layer are placed on the client, then object-based methods are used by the presentation layer to interact with them. The following guiding principle should be considered when designing a communication strategy:

- When online processing is not needed then offline processing should be enabled for the application. The connection state must be detected and monitored. When disconnected from the services then information should be cached locally and when communication is re-enabled then it should be resynchronized. For allowing shutdown-restart cycles and disconnected start up and of course it should be done without information loss, application state should be held and data persistent cached locally.

- When communicating with services then coarse-grained interfaces should be used to minimize network traffic and maximize performance. As we know that services reside on a remote physical tier so message-based protocol should be used whenever possible. If we want to avoid blocking the presentation layer, then following this way provides us a
more expected way to make asynchronous calls to. It also supports load balancing and escapes by failing server configurations.

- More efficient communication protocols for example TCP should be used for compression mechanisms so that data payload size should be minimized for message-based protocols such as simple mail transfer protocol and SOAP. When the application does not need to support open communication standards or custom binary formats then these protocols should be used. The network impact and probable performance should be considered, if the application consume or send large amounts or sets of data.

- IPSec and SSL is the best way to secure the channel and to protect sensitive information. Encryption is used to protect data and digital signatures are used to detect data tampering.

3.9.3 Composition
Users can open and close forms as required, and if these forms loaded only when it is needed then application can reduce start up delays and maximize performance. Where the user interface consists of separate modules or forms loaded dynamically at run time, to maximize maintainability and extensibility of the application and for the most particularly where it exposes to a complex user interface as is common in many business scenarios. As described above, this approach is very useful when users need to open several forms to work with data in a range of special ways and perform particular tasks. It is also necessary to provide the touch of personalization for users, so that they can change the content and layout accordingly to their need to go well with their own necessities. Here are some guiding principles which can be considered when designing a composition approach-

- Abstraction patterns should be used whenever possible so that maintainability issues could be avoided. Dependencies between components are also crucial aspect here. Abstraction patterns also help in managing versioning of compostable components and auto update. Where composition is appropriate, an appropriate composition mechanism should be identified and composing views should be considered from reusable modular
3. Material and Methods

parts. For example, to build a view from modular, use the composite View pattern from atomic component parts. Built-in features or composition framework such as the practices and patterns composite client application guidance of development environment could be alternatively used for this task such as user controls or document panels.

- Based on functional specifications and requirements and on the basis of requirement the appropriate types of interface components should be identified. For example, mostly used components for this kind of development includes WPF forms, user controls, Windows Forms, Office-style documents or custom modules.

- For making up a composite interface, there must be supported communication between different presentation components and forms. This will improve testability and also reduce the coupling between these components. In this way decoupled communication techniques are implemented, for example, Subscribe, Publish or Command pattern.

- Within the interface users should be allowed to customize their layout of composed components, this is done by implementing the personalization.

- Implementation technology provides appropriate templates and data binding techniques which minimize and simplify the required code for each form that is used within a composed interface.

3.9.4 Configuration Management

Configuration information is required by rich client applications during execution or when this is loaded at startup. This configuration information may be user settings, network or connection information, common design and display settings or user interface business rules. Changes to the information should be persisting when the application is running or when it ends. For example layout settings, user preferences, and other data is stored in the user's local profile. This kind of information (partial or complete) can be stored locally at client side or can be downloaded from a
remote server at the time when the application starts. When configuration management strategy is designed, the following guiding principle should be considered:

- Global application settings can be downloaded locally for performance reasons and this information should be stored at a central location. The place which is chosen to store this information should be local or centralized. Profile information or personalization settings which are used to manage user data should generally be stored locally. And if this data is stored centrally it means we want to enable roaming facility.

- The application should be designed in such a way so that it will be able to detect and dynamically apply the configuration changes, if it is needed. In this way we can come to know that what configurable data may change at some point in the life of our application. Here are some examples of these configurable data changes file locations, logging, assembly references, developer versus production settings, and contact information for notifications.

- A suitable mechanism should be implemented and sensitive configuration information should be identified. All these activities help in protecting it when persisted locally, during transit over the network, and even when stored in memory.

### 3.9.5 Data Access

For providing good application features, rich client applications need to access data stored on the local machine sometimes, as well as data stored on a remote server. Data transmission across tiers and performance of data access routines should be aimed to maximize as we know that data access has a major factor for the performance of the application. And it is the most significant impact in the user's observation of an application and it is also considered for its responsiveness and usability. So the types of data which are used by the application should be considered at the time of the designing of the application. However, this will be going to be a great factor for performance. A translation mechanism must be implemented which is able to convert it in the
3. Material and Methods

case if the client application is not able to handle the data in the exposed format. When a data access strategy is designed, the following guiding principles should be followed:

- It should be decided in the starting that how the inconsistencies between the original data and the local copy will be handled. Methods such as time stamps or events should be designed at earlier stage. If the client will consume very large amounts of data then to improve performance it should be loaded asynchronously into a local cache and should be chunked.

- Data should be loaded asynchronously, so that user interface can remain still responsive at the time when the data is loading. The interface should also be designed in such a way that it can protect against errors which arises at the time when user try to interact with this data before the completion of the loading.

- As we know that when multiple users attempt to update the central data store then concurrency conflicts arises. To overcome and manage these concurrency conflicts, pessimistic and optimistic concurrency models should be considered.

- For supporting batch processing, connectivity and implement a service dispatcher mechanism should be monitored so that users is allowed to perform multiple updates to data.

3.9.6 Exception Management

As we all know that all services and applications are having some problems so these are prone to errors and exceptions. To overcome from these errors and exceptions, a suitable strategy must be implemented. A robust and well designed exception management strategy can be used to detect and manage these errors and exceptions. And by this way security and manageability can be improved and application design is also simplified. If this strategy is used then it helps in reducing development time and cost and also makes it easier for developers to create the application. In addition, logging errors and exceptions should be considered for monitoring.
systems and use by operations staff. But this thing can be happen for other than validation messages which are used by trivial user interface errors. Usually it is needed to notify the user in a rich client application. One thing is more important here to design a centralized server-based logging sink and side by side collect log information so that the information can be accessed by all clients. While designing an exception management strategy, the following guiding principle should be followed-

- An overall strategy should be followed for handling exceptions. The actions which are considered for deciding this strategy includes wrapping exceptions with other application specific, replacing exceptions to prevent exposure of sensitive information and custom exceptions that contain additional data to assist in resolving failures. Beside all this, a mechanism for detecting and logging unhandled exceptions should also be considered. A framework such as the patterns & practices Enterprise Library can be used for the management of the exceptions.

- Some errors and exceptions are required the user notification whereas other do not. So the errors and exceptions that are probable arise within the application should be identified. Some errors usually just need to notify locally to the user like validation failures. But some errors like detection of malicious data and repeated invalid logon attempts must be logged on the server and must be notified to administrators. Although there are some executions exceptions should be logged but they are not necessarily notified to administrators like application failures.

- Only those exceptions should be caught that can be handled. Exceptions should never be used to control business logic. For example, data conversion exceptions can be caught when they are trying to change null values.

- All the things regarding exception handling should be determined as how to store exception information, how will it be notified to administrators and how this will pass to
other layers of the application if there is some need of this kind. A monitoring tool or environment that can read events from the local machine can be used so that present view of the application state can be provided to administrators.

- Exception information which is having sensitive information should be cleaned as exposing of this information to the users can cause problems. So it should be prevented from storing this receptive information in log and audit files and from being displayed. This can be achieved by encrypting the information and by using secure channels for the communication to other substantial tiers of that particular application.

3.9.7 Maintainability
For the development of the application the mechanisms should be implemented in such a way which could reduce the maintenance liabilities. It is fundamental need to reduce the effort for all applications and components and to minimize the maintenance cost. The design patterns which provide good separation of concerns and loose coupling between components should be used. Rich client applications are more difficult to operate than server installed applications and beside this, these applications are usually located on remote client machines. Some other issues are also there in maintainability are patches, versioning, deployment and updates. While designing a maintainability strategy, the following guiding principle should be followed-

- Based on the environment, an appropriate deployment approach should be used for the development of the application. For example, we can use system tools like Microsoft System Center which helps in deployment of the applications within an environment which is closed, or we can use an installation program for applications that are publically available.

- It should be checked and ensured that the application has interoperable and consistent versions of all the components which it is going to be use. Manual or automatic updates should also be implemented for to the application and all its components.
To solve run-time problems and to assist administrators and developers when debugging, logging and auditing should be implemented in an appropriate way.

Applications should be designed in such a way that components should be loosely coupled and should be interchangeable whenever needed. This helps in changing individual components whenever it is required; it also depends upon run-time scenarios, and personal user preferences. Also, components can be used in different scenarios where appropriate and should be designed to minimize dependencies between components and layers.

### 3.9.8 Presentation Layer

For the satisfaction of the user, some general factors such as design, interactivity usability and performance should be achieved by application. The presentation layer is the interface between the user and the application, this is the part the user sees and interacts with so it must therefore satisfy these all requirements. User experience is influenced by many special aspects of the application's architecture, so it is significant to design the application which can support in a better way and compelling and intuitive user experience from the outset.

- Investigate how we can separate the logic for managing user interaction from the user interface, and from the data with which the user works—perhaps by applying a Separated Presentation style. This makes it easier to update parts of the application, allows developers and designers to work separately on the components, and improves testability should be improved.

- Data binding capabilities provides the advantages for displaying data whenever possible, especially it is helpful for tabular data and multi row data representation. Data is automatically synchronized in different views or forms when these capabilities have been used. This also helps in application reducing the code required, reduces coding errors and simplifies the development process. Two-way binding can be used where the user wants the updating data.
3. Material and Methods

- Appropriate support for accessibility and navigation should be provided. And the application user interface should also fulfill the internationalized standard and then it should cooperate with all geological and educational scenarios and localized where it is going to be used. This provides the good user experience as it includes text direction, changing the language, auto detection of the user’s culture, and content layout which is based on configuration of the application.

- Style interface should be considered as it describes that how the documents are going to be displayed in an Office document or that how the document content or HTML are going to be displayed in other user interface elements. Beside the display of document the protection from the invalid and malicious content is also crucial aspect here as these things might also be present in documents.

3.9.9 State Management
State management is also one of the crucial aspects as it is responsible for the determination of the value of data. It shows the state of a operation, component, or steps of the process. State data can have configuration information, user settings, business rule values, workflow information, and data which is displayed by user interface. The application must be capable to handle conflicts, save data, restarts and access it as required. While designing a state-management strategy, the following guiding principle should be followed-

- A local disk based method should be used to store it if there is a large volume of state data, consider using to store it. And a persistent mechanism should be used if the application needs the data to be available at the time of starts up. In persistent mechanism data is stored at a disk file or isolated storage.

- The state information should be determined as the application stores it. It includes the frequency of changes, estimates of the size, and the overhead and processing cost of re-fetching or re-creating the data. State information also ensures that an appropriate support is provided by this mechanism.
3. Material and Methods

- This should also be considered that state information must be maintained at what granularity. For example, sometimes the state information is applied only to particular roles or users and sometimes it is applied to all users who are associated with the information and the application.

- Encryption and digital signatures should be used to provide the protection at an appropriate level when sensitive data is stored.

### 3.9.10 Workflow

For enabling of wizard style user interface elements or workflow or view flow support, rich client applications require multistep operations. These features can be implemented by using custom solutions or separate components. The advantages of a framework like Windows Workflow Foundation can also be taken for this kind of work. While designing a workflow strategy, the following guiding principle should be followed-

- While designing a workflow strategy, it should be considered that how the errors can be captured, managed, and displayed in workflows. Some features also considered here as in the case of a failure how the recovery is possible and in such a case what should be done like the process should be restarted or the task should be continued and how to handle the tasks which are partially completed.

- A workflow engine such as WF can be used for more complex view flow and workflow requirements. Custom code which is based upon well-known patterns is generally sufficient for this kind of requirement.

- Workflow can be useful for business components which are having operations that engage long-running or multistep processes. For implementing view flow and workflow tasks, separate components should be created. By using these separate components it makes easier to interchange components, reduces dependencies when there is some change in the requirements.
3.9.11 Security Considerations

Security includes a variety of factors and it is very important for all kinds of applications. And as we know that Rich client applications have a lot of its part at client side so security is a crucial aspect here. So security should be kept in mind when these applications designed or implemented. Beside this, other layers of the application should also be protected and secured as these application works as the presentation layer for business application. Security issues have a lot of concerns, these include user authorization and authentication, protecting sensitive data, protects against attack from malicious users and code, and auditing the user’s activities and logging events. While designing a security strategy, the following guiding principle should be followed-

- A number of approaches and technologies are there for user authentication. So appropriate approach should be selected for this work based on the requirements and security aspects of the user. This is also very important if the same rich client application is used by multiple users. There are a lot of things which should be considered when there are a lot of users such as when should users log on, different roles and types of users supported by application, differing kinds of permissions, and the recording of failed and successful logons. Offline or disconnected authentication can also be used where this is applicable without problems.

- Windows incorporated authentication solution or merged authentication services can also be used if the users need to access several applications with using the same identity or credentials. Beside Windows Integrated Authentication services a number of external agencies provide federated authentication solution. Certificate based systems, or creating a custom support for particular organization can also be used for the same purpose.

- In rich client application or any application the crucial thing is to protect the data which is stored in it and in respective resources like caches, files, and documents which are using by the application. Encryption and digital signature can be used to protect data.
Encryption is used for sensitive data at the places where it may be exposed, and digital signature is used to prevent tampering of the data. Encryption is used in most of the security applications in case of volatile information. Most important thing which should be considered is protection of sensitive information while it is sent over a network from the application.

- If we want the correct data should be stored in the application then it is also necessary that this data should also be correct when it is received by the application. So there is need for validating inputs received from both the user and sources like other application interfaces and services. This validation process of inputs can be done in two ways. We can use validation features which are provided by already developed technology or we might need creating custom validation methods or systems as per the requirement of the application. Some development environment like Microsoft Visual Studio Windows Forms makes available validation controls. Some third party validation frameworks are also available there like Application Block, Enterprise Library Validation; they provide complete features for the confirmation in the business layer and in the user interface. All the data should always be checked as any of this data can cross the trust boundaries.

- The implementation of audit and log is very important for the application, and it should be carefully decided that what information should be included in these logs. And as stated above sensitive information in the logs should always be protected by using encryption, and digital signatures can also be used optionally if there is some most sensitive information as it is exposed to tampering.

3.9.12 Data Handling Considerations
As we know that there are some web services which make the application data available receiving this from server-side applications. After receiving this data at client side, this data is cached there to get the better performance of the application and facilitate offline usage of this
data. In the same manner, Rich client applications also use the local data stored at client side. The data which is used by rich client application can be categories in two types:

- **Read-only reference data** - This kind of data is consistent means it does not modify often and client use this data for reference reasons, for example a product catalog. As reference data is stored on the client reduces the quantity of data which is exchanged between the server and client. By this way it improves the performance of the application as well as enables offline capabilities. Beside all these advantages it improves the usability of the application and provides early data validation.

- **Transient data** - Unlike the Read-only reference data, this kind of data can be altered on both sides’ client as well as the server. One of the most difficult aspects with transient data is facing the problem of concurrency issues which arises when same data get modified by a number of clients at the identical time. For resolving this problem, there must be a way to keep track of each and every client-side change in transient data on the client and on the behalf of this tracking information it should be managed and updated on the server side that may hold inconsistent or conflict changes.

### 3.9.13 Caching Data

Both the transient data and read-only reference data must be cached locally by the rich client application. As by caching data locally on the client side can improve the performance of the application and provides the data offline whenever necessary to work. So to enable data caching, some type of caching infrastructure is needed to implement by rich client applications that can handle and manage the data caching with transparency. Here are some general kinds of caching listed below-

- **Short term caching**. In this kind of caching data is not constant. That’s why this data cannot be used offline (when there is no connection with the server) by the application.
• **Long term caching.** In this kind of caching data is constant. And if the data is cached in a persistent means, for example local file system or isolated storage, it allows working offline also means application can run fine when there is no connection with the server. But for this application should be able to distinguish between data that is still tentative and data that has been effectively synchronized or coordinated with the server.

### 3.9.14 Data Concurrency

When multiple clients work on the application simultaneously then it is possible that changes in data resides on the server can take place before a particular client's changes get synchronized with the server data. This kind of things leads to data inconsistencies or corruption. So a mechanism should be implemented here to ensure that at the time of data synchronized, if there is any data conflicts found then it is handled appropriately, and the resulting data will be in consistent and correct form. Here are some general kinds of approaches which are used to handle data concurrency listed below-

- **Optimistic concurrency**- This type of concurrency supposes that the threat of data inconsistency is low. As the result, the data is not protected or locked by a client when it is going to be updated with optimistic concurrency. And for the detection of data conflicts, the changed data and the original data are both are thrown to the server. The original data which is supplied by the client is checked in opposition to the current data which is stored on the server to check if it has been altered after it was last retrieved by the client. And if the both values are same the changes applied. And in the second condition means the values are not same then a data conflict exception get raised by the server. This process of checking data is also known as optimistic offline locking.

- **Pessimistic concurrency**- This type of concurrency supposes that the threat of a data inconsistency is high. So to prevent data inconsistency, locking pattern is used. In this technique a lock is maintained over the data by the client, and this lock thereby stops any other clients to access or modify the data until this lock is released means changes made
by the client are finished and committed. This process of checking data is also known as pessimistic offline locking.

**3.9.15 Data Binding**

Data binding is mostly used for displaying read-only data to clients, which allows them to update or alter data within the user interface provided by the server. Windows Forms, Silverlight data binding and WPF support bidirectional binding which is used to bind user interface component with a data structure. In this way it allows the user to display the current data values, automatically updating to the underlying data with the help of values provided by the user, editing the data.

**3.9.16 Offline Connected Considerations**

An application is called occasionally connected if it cannot be able to interact with data or services with the server over a network during unspecified periods. These occasionally connected applications can perform their work when not they are connected with the server by a network resource, and can do the alteration work in the background when it is able to connect. At the time of designing the applications which are occasionally connected, the main aim resides on asynchronous communication as it interacts with services and data over a network only when it is needed and by this way it minimize or reduces complex interactions. This helps it to make easier the implementation of a synchronization mechanism of the communication for use when a connection to the server is available.

So it is easy to understand that these application interacts with the server when there is a connection available but in order to work when there is no connection, application should have data caching abilities to provide the storage of all of the data necessary on the client so that user can work continuously while offline. The application should also be prevented from using stale data in order to provide correct information. For this purpose a store and forward mechanism is used. In this technique messages or data are created, stored when there is no connection, and finally forwarded to the server at their final destinations whenever a connection is available.
Message queue is the most common technique of store and forward. Here are two general kinds of approaches which are used when designing a scenario of occasionally connected listed below-

- **Service oriented**- In the service-oriented approach the application stores the information in message form, and then arranges these messages in queue when it is not connected to the server and waits for the connection reestablishment. And when it gets the connection the messages which are waiting in a queue send to the server for storing and processing purpose.

- **Data centric**- In the data centric strategy the application uses a RDBMS which is installed on the client locally. Application uses here built-in abilities of the database so that it can handle the synchronization method; spread local data changes reverse to the server, and detect as well as resolve any data inconsistency.

### 3.10. CREATE OUR FIRST RCP APPLICATION

Here is an example of creating a simple RCP application so that an idea of Rich Client Platform and plug in concepts get clear. But for this application there should be some knowledge of uses of the Eclipse IDE in regard of the standard Java development.

#### 3.10.1. Creating an RCP Application

In Eclipse, first of all select File menu then New Project. From the given list select "Plug-In Project".
Figure 3.3: RCP Project -New Dialog Box

Now name your plugin "first".
Figure 3.4: RCP Project - Inserting Project Name

After that press "Next" and do some of the following settings. Select "Yes" for the question which is appearing in the dialog box—Would you like to create rich client application, as we are going to develop a RCP application,
3. Material and Methods

Figure 3.5: RCP Project - Making a Rich Client Application

After this press next and from the tab-available template, select the template "Hello RCP".
At the end of this dialog press next and select "Add branding" and finally press Finish.

As a result of this work, a project will be created having the features of the Rich Client Platform. The project structure is clear in the java source file of the project [34].
3. Material and Methods

3.10.2. Start your RCP application

For running the first RCP application, open the file "MANIFEST.MF". There will be an editor and by default the tab "Overview" will be selected. Click at the link "Launch an Eclipse Application" in this tab. And first running screen should be appearing on the screen like this:

Figure 3.8: Showing first Screen of Application
3.11 HOW TO RUN JAVA PROGRAM OUTSIDE ECLIPSE

3.11.1. Creating of jar file

Sometimes it is needed to run program of eclipse like java programs outside of the eclipse ide. This is also possible, but for this it is needed to export the required java program, as a .jar file. The standard distribution format for the java applications is basically .jar file.

Steps for creating the .jar file:

• First of all select the project for which we want to create .jar file. Then select the export menu entry from pop-up menu by right-clicking on it.

• Then from showing options select jar file, select next. Select your project and provide the name for the jar file and also maintain the export destination.

• Finally press finish. By doing this, it will create a jar file which will be saved in selected output directory.
This is the simple way to create jar file of java project. This can be run from the outside the Eclipse IDE.

### 3.11.2. Steps for running your program outside eclipse

Till now we have created the jar file but now the question is how to run this jar file from the outside of the Eclipse IDE. For this, open a command shell that can be found under Microsoft windows. Then select run from start menu and type cmd and press enter, console will be there. Now by typing cd path switch to that particular output directory where the .jar file resides. To make to run this program, the jar file should be included into classpath. The classpath have both the information that which java classes are available to the java runtime and where they can be found also. We can add a jar file to the classpath with using the -jar option. Here is the example of it-

```
Java -classpath myRpcApp.jar First.myownclass
```

As soon as we type the above command, it will provide the output of the program in the directory which is used for the export.

### 3.12 REFACTORING IN ECLIPSE

Now days, a process model is used to develop most object-oriented software systems. Changes are needed in evolutionary development lifecycle time to time. One of the important changes in the object-oriented software is Refactoring. The main objective of refactoring is improving the worth of the software structure which is a crucial aspect of any software. It improves the maintainability, extensibility, and understandability of the designed system without affecting its overall behavior and functionality. If we want some changes in the system without changing its behavior then we have to restructuring the code and it should not affect the behavior of that system, Refactoring help us to do this. Renaming a Java method or class is an example of Eclipse as it supports the refactoring activity, supports moving or renaming.
The way to use the Rename refactoring is right clicking on class which can be found in editor or Package Explorer and then selecting the Rename from Refactor tab to rename the class. Eclipse will automatically change the all calls to the method or class is renamed in the Workspace.

Figure 3.10: Refactoring

Good modularity can improve the Quality of Software. It facilitates independent development of components, extensibility and evolution, eases verification, improves comprehensibility. The value of the modularity is computable. A developer may have a number of concerns about the implementation of a software system. Good modularity means that software is enriched with a small number of concerns which is implemented by each module; each concern is implemented in only one module. If this kind of structure follows, then complexity is easily managed by developer. Complexity can be removed because only having a look at one module is enough to deal with any one concern and to understand the complete one module developer have to consider only about a small number of concerns.
Constant evolution and change are required by any useful software system. For the system re-modularized some changes are required. This is needed so that the system becomes easier to maintain, understand, extend. For this type of need, developers and researchers do the practice of refactoring. We can describe refactoring as parameterized transformations of a system’s source code so that it can improve the system’s structure with regards to informally expressed goals, such as performance, maintainability, readability, changeability, and memory demands. Built-in support for semi-automated refactoring is included by modern software development environments.

Through using integrated development environments such as Refactoring Browser and Eclipse, available for semiautomatic support for low level refactoring, such as Move Method and Rename Field. Integrated Development Environments is also support improved code quality and convert it less bulky. This is the beneficial impact of software Design.

Some activities are more difficult such reuse-based development by using Refactoring. If changes are require for running software at same time it depends upon its design and implementation of components that are used by software. Their activity has to be limited to changing the internal implementation of the application refactor application code. Application programming interfaces are used to reusable components and it cannot change or remove existing parts of application programming interface or it may fail the client side applications.

The defined framework used components for cost of contravention client code therefore cost is high, due to this the making changes improve the components quality. During refactoring the potential challenges may occurred. The context of reuse-based development give improves the need for understanding how this activity is actually worked. Regrettably, less amount of work has been done on investigating using refactoring, the lifecycle of object-oriented software system are refactoring. There is no substantial tool for classifying and detecting structural evolution for applying refactoring. The documentation, log of changes and release notes, reports are not available to study the actual changes.
Thus, numerous important questions stay unexplained-

- Which kind of support should modern IDEs provide and in what way they support it?
- Which type of refactoring applied in practice?
- In what quantity of the structural changes are the results of refactoring?
- On what kind of systems these refactoring systems are safe to client applications?
- How the system’s structural evolution can be gathered automatically?

Eclipse is basically a plugin-based framework. Eclipse is a large-scale industrial framework that is continuously growing and developing. In this process of developing, it has obtained a large user base and a number of different applications have been built on it. It provides a lot of ease of development so its users can use it as an IDE as well as they can also extend or build their own plug-ins. This plug-ins can be developed from core from the existing ones. From the version 3.0, a concept of a rich client platform is also introduced by Eclipse which helps its users to build stand-alone applications from a subset of plugins. As a result, Eclipse can help us understand the design requirements for refactoring-based development environment. And it also helps in studying the structural evolution of from the perspectives of both the component developers and component users.

The term refactoring was introduced by both Griswold and Opdyke virtually at the same time. For automated refactoring, they provided the theoretical basis of the work through refactoring tools. Eclipse offers the way by which it can support refactoring. By providing a catalogue of refactoring, Fowler makes the Eclipse popular.

Refactoring became an essential part of the evolutionary software development methodology. Object oriented programming is the main concentration of any research so most refactoring research is done in the field of the targeted low-level program. This can transform the functional and object-oriented systems. Fowler and Kerievsky have done a lot of work in the field of the refactoring. So they present a lot of information about the refactoring in their book, beside this,
they have explained all the things about it as how they can be used to carry out architectural and design changes.

Modern Integrated Development Environment, like Eclipse, provides support for refactoring, so in this way it provides features such as move and rename. Modern IDEs do not supply automated support for refactoring which is frequently used like high-level refactoring. Although refactoring is very good concept for reusing the components but it is having some drawbacks also as sometimes it can cause breaking the client code. When this type of things happens, there is the burden of migration of the codes to the new version for the developers who are developing the component-based applications.

Beside all these features, Eclipse also provides support for moving methods and static fields to a specified type. But in this case it treats moving instance fields just like a textual move and that’s why when these instance fields moved, the references of the moved instance fields remains same, they will not be updated.

As above paragraphs explains that Modern IDEs support the most commonly used, low-level refactoring. But downcast type and information hiding types of refactoring is not supported by these modern IDEs. As in information hiding refactoring, a class has to hide several members and this is not so easy task. And if is done manually, it could be error-prone.

If we talk about Eclipse, it supports moving instance methods of only those types which are declared in the same class as the method only. Moving instant fields and methods to their direct super class or subclass is also provided by Eclipse. This kind of refactoring is known as ‘push down’ or ‘pull up’ refactoring. Beside all these kind of refactoring, Eclipse supports some of the refactoring which are bigger, but these types of refactoring lack support for the refactoring of the general class relationships and containment and inheritance hierarchy.

Although small, primitive refactoring are powerful refactoring but using the bigger refactoring like by combining the relevant low-level changes into composite high-level refactoring, the
developing work becomes more efficient and by this way it can implement the specific intent of the change. As on basis of previously research about refactoring which is provided by Eclipse, it works all the way through its development history. In addition to what are commonly supported in current IDEs, an effective refactoring tool should support the following features-

- It should provide a refactoring user interface which can collect the information about more complex refactoring tasks. This happens in inheritance hierarchy refactoring.
- In the case of entity which is object-oriented should be more flexible at the time of moving of instance field and method instead of simply text.
- It should support in hiding of a group of method in a class as it happens in information hiding refactoring.

3.13. OPEN SERVICES GATEWAY INITIATIVE (OSGI)

To deliver products and services to end-users, service providers use the gateway. This is majorly used in home security or health care monitoring kind of products. For use in residential gateways, the OSGi specification defines a service-oriented framework. In this context, the OSGi framework does the role of a gateway between the Internet and residence’s home-area network’s consumer device. The devices themselves become mechanisms for delivering services to the framework as more powerful consumer devices are introduced day by day. When all these features are provided it results into complex applications. And then the need arises for providing the facility to the developer so that he can develop it in simple manner. Here is the description of the path from the proprietary Eclipse 2.1 runtime to the new Eclipse 3.0 runtime based on OSGi specification. It details the motivation for such a change and discusses the challenges this change presented.

The OSGi Alliance is an open standardization organization which is formed in March 1999 combined by International Business Machines, Sun Microsystems, Ericsson and others. It has specified a Java programming language based service platform over the past few years. It defines
an application life cycle model and a service registry. OSGi can be remotely managed. The following OSGi services have been defined and used by developers:

- Package Admin
- Configuration Management
- IO Connector
- HttpService, used to run servlets
- Wire Admin
- XML Parsing
- UPnP
- Preferences
- Jini
- Log
- Start Level
- Device Access
- User Admin
- Permission Admin

The framework is a dynamic and complete component model. The policies of life cycle management can be downloaded and applied. One can start, install, stop, update and uninstall applications from a remote location. Bundles were distinguished by the service registry as they are applications that can adapt to new services. In the starting days the original focus of OSGI was upon providing the service gateway, but nowadays it growing as it is used from mobile phones to the new version of the Eclipse IDE.

The Open Services Gateway initiative framework is a service platform and module system for the Java development that implements a dynamic and complete component model. This kind of model does not exist before 2011 in separate Java virtual machine environments. Components or
Applications which are basically in the form of bundles can be used remotely. The installation, starting, stopping, updating and uninstallation of these bundles can be done remotely without requiring rebooting the machine. So in this way it provides the help in management of Java packages, classes specified in great detail. Beside all these things the service registry allows bundles to identify the accumulation of new services, or the elimination of services if there is any and adapt these services accordingly. Application programming interfaces that allow for remote downloading of management policies also handles the application life cycle management. So it is responsible for starting, stopping, and installing it.

Of course, consumer devices are more integrated into everyday life and are much more prolific than personal computers. For software developers, it is necessary to understand consumer device technology which provides an more and more interesting environment in which they can develop and deliver products and services. By accessing the user by affecting their products and services towards consumer devices, software developers get benefits. The following are four aspects of consumer device software market-

1. The renovation of traditional consumer devices into smart consumer devices like refrigerators and washing machines.
2. By the increasing the features and power of consumer devices like personal digital assistants (PDA) and mobile phones.
3. The increasing popularity of home area networks.
4. The convenience of broadband Internet connectivity which is widespread.

The OSGi makes developers realize the prospective of the consumer device market. The OSGi specification defines a service platform. This service platform is a small framework for managing the components and it comprises a minimal component like model and a small framework which is able to manage the components, including a packaging and delivery format. Open Services Gateway Initiative (OSGi) is an independent and non-profit corporation which promotes open specifications and defines all the working of it. OSGi is responsible for the
delivery of managed broadband services to home networks as well as car and other environment networks.

A service is nothing but a Java interface with hypothetically multiple implementations and defined semantics. Services are deployed via the internet into the OSGi home services gateway and packaged along with their associated resources. The OSGi framework facilitates controlled simple components known as bundles. These bundles are dynamically loadable components. They can interact with each other, provide services and in this way are able to form applications. In this situation, consumer devices which are capable to deliver bundles become delivery mechanisms.

If a typical application scenario is concerned like a home security company, the main work of this company is to monitor the end-user's home for security aspects. For this purpose it uses a combination of software like software control panels, sensor monitors for the end-users and the service provider and hardware like sensors. Working with this scenario, the services gateway acts as both the channel and the access mechanism. This service gateway works between service providers and the end-user. On top of the OSGi framework the low abstraction level increases the difficulty of contribution complex applications at the time when OSGi creates a good foundation on which to build such services.
3. Material and Methods

Figure 3.11: OSGi Framework

The OSGi specifications have lost its original focus of service gateways. And now they are used in applications ranging from open source Eclipse to the IDE mobile phones. The OSGi specifications are now using for many different application like industrial automation, automobiles, building automation, grid computing, PDAs, entertainment, application servers and fleet management.

3.13.1. The OSGi Service Platform

The OSGi Service Platform has two parts mainly-

1. OSGi framework
2. OSGi standard services

Since the framework defines what a service is, it is most important. Specific services and their specified functionality are defined by the standard services. Services which are provided by OSGi are simply Java interface definitions which have precise and specified semantics. any
object that implements a service interface and the creator of the service interface defines the semantics of a service interface.

A bundle is a Java JAR file that contains a combination of Java class files, manifest, native code, and any associated resources. By using host environment, the OSGi framework manages the services and bundles they provide. A bundle is a logical concept used by the framework and it is a physical unit of deployment in OSGi and is also for organizing its internal state. An installed bundle is uniquely identifiable in the framework by two ways-

1. By its location- This can be arbitrary character string used at the time of the installment of the bundle.
2. By its bundle identifier-This is a number which is assigned dynamically by the framework at the time of the installment of the bundle.

Since bundles are uniquely identified by their location string, in common practice the location string of a bundle is typically a URL, but the specification does not define it as such. Bundle activation, installation, deactivation, removal and update all these features are provided by management mechanisms. Reasonably, installed bundles have one of the related states at any given time. The value of these states can be one of the following values-

- Installed
- Starting
- Resolved
- Stopping
- Active
- Uninstalled

When the framework is shutdown, active bundles are returned to the active state or we can say that the state of active bundles is persistent across the framework activations.
The manifest file corresponds to the entry in the JAR archive named META-INF/MANIFEST.MF. The bundle to JAR file mapping is one to one and the JAR file, metadata about the bundle is available. The standard manifest attributes describe native library requirements, the class path for the bundle, any imported Java packages required by the bundle, any exported Java packages provided by the bundle and information intended for humans. The manifest file is simply a group of attribute value pairs and some of these attributes are standardized by the OSGi specification.

Since it is the mechanism by which bundles are able to get a context for accessing framework functionality, the activator class plays an important role. Besides the abovementioned attributes, the manifest file may also contain an attribute which is used to specify an activator class for the bundle.

When the bundle is started or stopped, the activator class is called. And at each of these instances, for accessing the framework the bundle is given its specific context. The context allows the activator or any other class that has access to the context to register services of its own, install additional bundles, look up services in the framework’s service registry, and access other bundles. If there is a bundle which is only a library of Java packages and it does not need to access the framework, then in this case bundles are not required to have an activator class.

The defining bundle export the service interface definitions so that other bundles that want to use the service can easily import them. And the service implementations remain private also. Services are implemented by a separate Java class and are specified as a Java interface and its associated classes and resources. These all things are packaged as a bundle for example a JAR file with a manifest file.

A bundle implementation may register services to be used by other bundles, may request services from other bundles, or it can do both of them. The OSGi framework is a java framework that runs on the top of a java environment. In the java development environment Software
programmer can deploy new applications as well as can write new applications to run on the OSGi framework. Up to now, the OSGi framework has implemented the Log service, HTTP service, Device Access Service and the core framework of the OSGi specification. To support applications that consent new application or services to be deployed during the life of the device, the framework extends the Java runtime environment.

3.13.2. OSGi Limitations
The OSGi specification does not significant for a sophisticated component model which is necessary for developing complex applications. The main focus of the OSGi specification is focused on defining a services gateway.

3.13.3. Dependencies
There are mainly three types of dependencies present in OSGi-

1. Bundle-to-package
2. Bundle-to-service
3. Service-to-service

- **Bundle-to-package**: Classes which resides in a bundle may need to import code that is External to their JAR file. In this kind of situation, the bundle must explicitly import the code. Package dependencies are declared at manifest file. Other bundles can explicitly export Java packages, if there is need to satisfy import requirements. And if there a case that the framework is unable to fulfill a bundle’s import requirements, then it is not possible to start the importing bundle.

- **Bundle-to-service**: In the framework service registry, classes inside of a bundle may use registered services by querying for them. Service dependencies, as stated above, may be declared in manifest file, but this is just for information purposes only. Service dependencies are not
guaranteed by the framework unlike the package dependencies. This means bundle to package dependency, a bundle can be started even though the services that it requires are not available.

- **Service-to-service**: In the OSGi framework Bundle-to-package dependencies are handled in a well manner so that it provides simple version control where as service-to-service dependencies are not covered by the OSGi specification. Both the dependencies bundle-to-service and service-to-service do not handle by the OSGi framework, even if bundle-to-service dependencies are described in the bundle manifest. A bundle can have numerous service implementations, some of them have satisfied their dependencies and some of them do not.

  Since both bundle-to-service and service-to-service dependencies are not externally manageable, they are implicit in OSGi. Instead, the details of these dependencies are suppressed inside of their respective service implementations and bundle. As the results of this OSGi applications are difficult to manage and deploy.

  Services are only interfaces and that’s why they do not have dependencies themselves as do not specify an implementation. In other words, a service-to-service dependency cannot be explained as dependency between two services; rather than it can be understand as just a dependency between a specific service implementation and another service interface. It is not possible to allow the user to externally configure service dependencies nor is it possible to create a service to automatically install all required services for a bundle if specific service implementation is wanted to use by developer. Bundle-to-package and bundle-to-service both the dependencies are impartially easy to understand, but service-to-service dependencies are not so much easy, these are a little bit complicated. There are multiple independent factors those tells the characteristics of service-to-service dependency. Two important factors of them are cardinality and dynamism. It is not there that implementations of a service can have same dependencies all times; it is possible that multiple implementations of the same service may have different dependencies. Cardinality which is one main factor for service-to-service dependency is valuable for expressing optionally, such as aggregation, one-to-many dependency and zero-to-
one dependency. Dynamism is used to indicate whether a dependency tracks changes at runtime in the environment, for example the arrival of new services which completely satisfy the dependency constraints.

Notice that bundle-to-bundle dependencies are essentially forbidden in OSGi, so they are not present there. This means the framework is free to resolve these issues as it sees fit and it is not possible to hang on the service or packages of a specific bundle.

3.13.4. Application-level

OSGi is not capable to handle dependencies among bundles and services or among services implementations although it is able to handles package dependencies. An application in the OSGi framework means that it is the transitive end of all dependencies among bundles, packages, and services. Because of all this, neither OSGi have an unambiguous notion of an application nor is it able to define any application-level services, for example automatic application deployment.

3.13.5. Complexity

Services may appear or disappear at any time; this is an important aspect of the service-oriented paradigm of OSGi. This means a set of bundles connected via service dependencies or an application must be able to handle this dynamic situation. Because of all this, all bundles must implement complex code so that they can handle both the binding and unbinding of services as they dynamically reach and leave.

3.14. 3D GRAPHICS FOR RCP WITH USING OPENGL

For developing interactive, portable 3D graphics applications, OpenGL has become the de facto standard environment. OpenGL is one of the most widely used and supported 3D graphics Application Programming Interface. Thousands of applications to a wide variety of computer platforms are brought to us with the help of OpenGL.
3.14.1 OpenGL

It was first introduced in 1992 and can be seen as a graphic library which is based on commercial graphical system by SGI and. It is worldwide known interface and is used in both industry and games for graphical output. Opposite of GDI (GDI+) it is aimed on performance and it makes possible for the user to use available features of installed graphical hardware. It provides support for displaying of 3D and 2D object.

![OpenGL Example](image)

Figure 3.12: OpenGL Example

3.14.1.1 2D Object Support

The main aim of OpenGL is visualization of 3D objects. Support for 2D objects is limited and it is based on 3D objects, i.e., drawing of line as 2D graphical primitive is similar to drawing of line as 3D object on a surface. It is also possible to use OpenGL for pure 2D operations like image processing. This is because of the interface provided by it as it contains functions to set and retrieve the values of pixels inside a specified area which places at the target frame buffer. Unluckily, the capabilities of these functions are not sufficient and these functions do not have
good hardware support also. This usually leads to lower quality of output; nearest-neighbor approach is used for scaling or performance loss. Therefore as these pure 2D objects are having so many problems, it is better to use simple 3D objects with texture mapping. Beside this, the features that are commonly supported by the hardware, the use of simple 3D object instead of 2D ones leads to improvement of performance.

3.14.1.2 Basic Features

OpenGL library provides complete rendering pipeline. This pipeline supports facilities for visualization of 3D objects, i.e., it handles clipping, lighting, texturing, transformations, and visibility solving. It may handle more tasks; however, these ones are fundamental and are supported by all versions of OpenGL. The given frame buffer or window of current Graphical user interface is getting rendered by the result of 3D world projection.

Each object is described by its surface. Geometry of such surface is easily specified using basic few render primitives, such as lines, points, quads including quad strips, triangles including triangle strips and fans, and. The library also provides light and lighting computation as the rendering pipeline’s standard. It is possible to choose the light from common light types, such as reflector, directional light and point light to set up their parameters. These light types are supported by the hardware often. So if this kind of support is not available on current graphic hardware, OpenGL implementations provide software emulation.

User-defined material can also be used to cover the surface of every rendered primitive.

The material describes interaction of surface and light, i.e., it describes its color. The description of surface may also include texture specification. Texture is an image that is mapped on the surface (i.e., surface is covered with the texture). Each object may be covered with 2D or 1D texture. Use of multiple textures on single object (i.e., multi-texturing) is also supported as well as techniques that shall improve the result of texture mapping (e.g., map mapping, perspective correct mapping).
3. Material and Methods

To allow object geometry manipulations, OpenGL provides support for transformations of rendered geometry (i.e., vertices) before its rasterization on target (screen, image) surface. Transformations are provided in form of matrices. OpenGL uses it and provides support for combining of simple (particular) transformations into complex one. The user can set the particular transformations in the form of matrix by standard library functions. A possibility to parameterize basic transformations is offered by such functions, such as scaling by given coefficients, translation by a given vector and rotation around given axis. The output of such functions is automatically combined with results of previous combinations.

Currently there are two matrices in OpenGL: projection matrix, which is used for final projection of transformed 3D objects on 2D surface, and model view matrix, which is used to transform rendered 3D objects before their projection and rasterization. User is allowed to retrieve, to paste, or to store current transformation (both projection and model view) matrix on internal stack. The latest mentioned feature simplifies rendering of hierarchical object (e.g., robot arm, human body, etc.).

OpenGL also provides support for solving visibility of rendered objects in form of Zbuffer. This approach of visibility solving is based on pixel basis, it is supported by common graphical hardware, and therefore its use does not decrease performance. Similar to Z-buffer OpenGL also supports stencil-buffer that is used to mask parts of render target on per pixel basis. However, stencil-buffer is not supported by older graphical hardware and therefore its use may lead to trivial recital deprivation in some cases.

3.14.1.3 Inside of OpenGL

The view behavior of OpenGL is similar to a state machine from inside. Each function modifies the current state of the machine. The functions used to retrieve state and data are excluded here. The result of the rendering is influenced by this state then modifies the function behavior also.
3. Material and Methods

Standardized specification describes the Interface functionality and behavior. However, a real implementation is not like the description which is found in the interface functionality. It is a little bit different. It basically trails the behavior described in the specification but in some cases it slightly differs like in error handling.

Very stable and robust background is provided by some implementations. So they can absorb user's mistakes and do not provide any visible feedback. While in the same case and such mistake others strictly follow the specifications and an unpredictable output is provided.

When user develops his/her application using robust implementation, it can lead to difficulties while debugging and user gets surprised as he gets strange output using another implementation which is less robust. This depends also on the used version of the device drivers and sometimes on the used graphical hardware.

3.14.1.4 Extensions and Other Libraries

OpenGL provides functionalities for rendering the objects of basic primitives with defined properties like texture, lights, etc. Unfortunately, this functionality is sometimes too difficult to use or not sufficient for the desired output. For solving such kind of problems, OpenGL works with some existing libraries like GLU, GLUT or some existing add-ons like GL Extensions.

One of this OpenGL Graphic System Utility Library is the GLU. It simplifies setting of projection transformation and tessellation of parametric surfaces like NURBS and quadrics like sphere provides facilities for rendering. It provides hi-level functionality and functions for OpenGL.

Another OpenGL Graphic System Utility Library is the GLUT. Its main aim is unification of OpenGL and simplification and initialization of OpenGL. It also cooperates with currently available Graphical user interface. One thing about the OpenGL should be cleared here that OpenGL does not contain any facilities for input because it handles output only. OpenGL
interface itself is standardized by specifications while its initialization and cooperation with current Graphical user interface is not standardized by specifications.

Due to that, the initialization may not be simple enough and it is as well as user input handling platform specific. Therefore, it may complicate porting of an application to another platform. The GLUT library unifies these tasks by providing environment for it and also makes source code portable so that it can run on different platforms.

GL Extensions creates its own set for each graphical hardware vendor, this creates the problem and that’s why these GL Extensions are not part of the specification. These GL Extensions are part of the OpenGL library. As these extensions become the part of library, they can use the latest hardware features although they are not part of specifications. But for this reason, OpenGL implementations cannot able to support their software emulation as they are not supported by graphical hardware.

3.14.2 Starting with OpenGL

The Java community got new heights when Java requisite to OpenGL as the result of the partnership of SGI and Sun particularly in the field of game developing, graphics and desktop. Though many developers get excited to see the popular and widely understood OpenGL exposed in a more direct fashion to Java developers yet some persons get disappointed to see Sun back away from Java3D.

Java/OpenGL binding is reference implementation which is hosted on java.net as the JOGL project. Some essential points of JOGL are listed below-

1. With using simple graphic primitives and images, how to do 2D graphics in JOGL.
2. How to create a JOGL powered AWT component which can provide good receive and respond to events like repaint requests and size changes.
3. How to provision the JOGL library files and download them.
We all know that 2D keeps things simple and 3D provide good graphics. OpenGL uses 3D so it also provides good graphics. But there is good idea that if hardware-accelerated 2D application programming interface works with JOGL then it can be used as a rendering pipeline for Java2D, then everyday uses of Java2D, including Swing, could gain a significant performance improvement. These things are also being done for Java 1.5 when used with Linux and Solaris.

Presently JOGL is supported by four platforms-

- Windows
- Mac OS X
- Solaris
- Linux (on x86)

For building the software on each platform, the JOGL site specifies minimum requirements, but not for running it. The build requirements specify Windows 2000 and Red Hat Linux; the run requirements are clearly more permissive. The preliminary user's guide mentions some important considerations which note that J2SE 1.4 is a minimum requirement on all platforms.

The basic requirements which are needed for JOGL's home page can be found in "Getting Started" section. All documentation link and precompiled binaries can also be found at the "Downloads" section. According to the platform the distribution can be downloaded and can be decompress on system. As there is some difference platform to platform, each compressed file expands into a jogl.jar file and in native shared library files- libjogl.so on Linux, libjogl_jnilib on Mac OS X and jogl.dll and jogl_cg.dll on Windows.

The entire .jar file like jogl.jar should be set as CLASSPATH for compiling and running the code. In the same manner the native library file or files should also be needed to be along the java-library-path at run time. For doing this, there are two options available, this are-
1. Place the .jar files in the lib/ext directory which can be found inside the Java home directory. And place the native libraries in an equivalent always-checked directory (such as jre/bin on Windows, or /Library/Java/Extensions on Mac OS X.

2. Here files can be uploaded with code and can access them directly with -classpath and -java.library.path arguments which can run on command line. And in this way it becomes easier to deploy the JOGL application. By this way, this approach helps the end users for adding the files to these directories.

3.14.3 standard widget toolkit and OpenGL

And when all these things work properly, one main question arises at this time that how RCP interacts with OpenGL. This interaction is made possible through standard widget toolkit with the help of two fundamental application programming interfaces-

A device-independent package- Thepackage: it uses two basic classes which support OpenGL. It integrates between standard widget toolkit applications and OpenGL graphics. The package integrates standard widget toolkit and OpenGL graphics by providing platform-independent OpenGL support through two classes.

An OpenGL binding: The OpenGL specification interacts with the machine hardware using this layer. Java Native Interface (JNI) is used to bind to perform native calls, and it is responsible for accessing the hardware, such as the graphics card.

Some Pixel Attributes specifies a double-buffered surface used to configure the OpenGL State Property Description. Pixel format attributes are in fact a set of properties that define the OpenGL state for drawing in preparation. A GL drawable is simply a 3D rendering surface. Graphics operations can cause the rendered images to appear to flicker or have an otherwise unacceptable appearance. These Graphics operations require multiple complex painting operations also.
All paint operations are first rendered to a memory buffer, instead of to the drawing surface on the screen, when double-buffering is enabled. The flicker problems which are associated with multiple paint operations are addressed by double-buffering by using a memory buffer. The memory buffer is copied directly to the drawing surface associated with it, when all paint operations have been completed. And as only one graphics operation is performed on the screen at a time, the image flickering is eliminated automatically.

Features of OpenGL 4.3 include:

- It eliminates the need for a different set of textures for each platform by using high quality ETC2 / EAC texture compression as a standard feature.

  For advanced computation like image, volume, and geometry processing in the context of graphics pipeline it computes shaders that bind GPU parallelism. By using of Shader storage buffer objects it enable vertex, geometry, tessellation, fragment & it also compute read and write large amount of data and transfer significant data between shader stages.

- Texture is able to view for understanding textures in many different ways but it need not duplicate the texture data itself.

- The texture parameter depends upon the current platform to query supported texture to discover actual parameter.

- Debugging messages are received during application development.

- The GPU are enabled by indirect multi-draw to store and compute parameters for multiple draw commands in a buffer object. It also reuses those parameters for efficient rendering with one draw command.

- An application that causes a GPU rearrange will not affect any other running applications. This is ensured by multi-application robustness extension.
3. Material and Methods

- It also Increases memory security in the way that an application cannot read or write outside its own buffers and cannot access another application’s data.

3.15. METHODOLOGY

3.15.1 Research Strategy
To gain an insight into the work of developers a number of plug-ins studies. The selection of the case study approach is done with the fact that it could explore particular cases, where web 2.0 applications were being developed and we had access to the developers themselves. The reason why it has chosen to involve in this research is because they are the by this way only I can get the problems which are faced in projects developed in Rich Client Platform. By conducting a case study we could identify problems and needs of developers in direct relation to their work, resulting in an in-depth study of particular cases (Creswell, 1998). The fact of designing an application within a company puts some restrictions onto developers: the company’s goals have to be fulfilled and still the application has to be usable and finished on schedule, even with limited resources for usability testing. Thus, the developers’ work is tied to its context, which calls for a case study to investigate it as a whole (Yin, 2003).

Aspects regarding development are generalized and included into discussions of a guideline outline.

3.15.2. Research Methods
3.15.2.1 Literature Review
The literature review presents a general picture of the Rich Client Platform Developments context and shows which work has already been done in the field of designing projects based on it. It helps us understanding the topic and focusing on the relevant parts before conducting the case study. For the readers, the literature review serves as a further introduction to the area, making them familiar with existing guidelines. The knowledge gained is crucial for designing this Earth Navigator using Rich Client Platform.

3.15.2.2 Case Study
We conducted a case study by analyzing the work of professional designers in various settings, include commercial & academic. The case study involves the background of the organizations the designers work for and employs interviews as a core data collection technique. These are an important information source, as they show the “real world” situation from the point of view of our target group.

After finishing the case studies, I analyze and summarize the main aspects needed for development in Rich Client Platform. I believe that selecting right case studies will get valuable results for the research.

3.15.2.3 Research Quality

In order to enhance credibility of my work, I use methods to enhance research validity and we put a strong emphasis on the ethical aspect of our research. As everybody is aware of the fact that literature presents a more theoretical point of view, whereas a case study will highlight practical aspects. Yet, if both data sources will generate similar outcome, then our research outcome will gain more validity. The basis of this approach is derived from the ethics of virtue and the simple fact that practically how this rich client platform helps user as well as developers. The central aspect of my work is finding out how creating Rich Internet Applications could beneficial. If the outcome of the research will benefit someone, it will affect web developers who can do their job better, and in consequence, all web users.

3.16. DESIGN

This section shows the architecture of the application and how parts of the application can be split up into own plug-ins.

3.16.1 Earth Navigator

The earth navigation consists of a series of tasks, which are interconnected. The framework that will be used consists of three main packages, as can be seen in figure 3.9. These packages are
views, handlers, and util. In Package view, there are three classes- LeftView - for Navigation View,
RightView - for Globe View,
YGeoSearch - for using the utility to get the longitudes and latitudes of particular area.

![Project Structure Diagram]

**Figure 3.13: Project Structure**

In the object-oriented community, the word framework has a technical definition. In simple terms, framework can be considered as design structure that can be reused by different components and structures. If we talk about simple code reuse, there is only some parts of the code are reused by other component or another functions whereas in the object-oriented framework data flow of the application is defined. Both the traditional code and framework uses the technology of reusing the code. But there is difference between these two techniques. In the
traditional reuse it involves inserting pre-existing elements into the particular structure whereas the framework involves inserting the particular elements into a pre-existing framework. In a design sense the other advantage of the framework is that the code can easily be tested, reused and debugged. The purpose of the Earth Navigator, a plug in which is designed for this research to know more about eclipse rich client platform, is to navigate the earth on Eclipse platform. For fulfilling this requirement, there is need for the well-defined framework first for the Earth navigator plug-in.

3.16.1.1 Building a 3D Earth Navigator on Eclipse RCP

Here for the purpose of this research; I have built a real-life application for Earth Navigation which is using the National Aeronautics and Space Administration (NASA) World Wind SDK and JOGL. The application which is developed here for Earth Navigation is similar to Google Earth. Following components are used in this application-

1. First of all it uses SDK. This SDK helps the developer to use the technology of World Wind’s geospatial visualization. This technology uses the NASA’s geospatial data which allows developers to embed technology in their own applications.

2. The Google Geocoding application programming interface, which is generally designed for geocoding static addresses for placement of application content on a map; this service is not designed to respond in real time to user input, for example. For dynamic geocoding (for example, within a user interface element), consult the documentation for the JavaScript application programming interface V2 Client Geocoder, the JavaScript application programming interface V3 Client Geocoder, or the Maps application programming interface for Flash Client Geocoder.

3. Geocoding is a time and resource intensive task. Whenever possible, pre-geocode known addresses (using the Geocoding application programming interface described here or another geocoding service), and store your results in a temporary cache of your own design.
4. Allows finding the specific latitude and longitude for an address.

Before going into deep in the application, it is necessary to review some WorldWind JAVA (WWJ) basics.

3.16.2 WorldWind JAVA Basics
A map tiling system is used by WWJ. WWJ also uses Cartesian coordinate system to divide the sphere in rectangular sections and by this way it shows images on top of it. The WWJ SDK provides the 3D graphics globe, which is built on top of JOGL. A latitude/longitude bounding box is part of each section.

For showing the images of earth, WWJ gets the images from NASA dataset servers and after fetching them WWJ projects these images onto the sphere. As the number of tiles can be huge at high resolutions, each tile is cached on disk of that particular computer for good performance. Whenever the user zooms in the images, the number of tiles quadruples but it depends on the computer zoom level.

Blue Marble is the most important dataset of WWJ. Blue Marble displays the images of the earth which it takes from the NASA’s imagery of the Earth.

3.16.3. World Wind System Architecture
The World Wind application programming interface mainly helps in replacing components with alternative components selectively for third-party developers. This application programming interface is defined by interfaces. WorldWindowGLCanvas is the main class of WWJ class hierarchy, which is again subclass of an AWT component.

Some major works of World Wind application programming interface are as following-

1. World Wind application programming interface represents a planet’s shape as well as its terrain.
2. World Wind application programming interface represents the highest level interface with the OpenGL canvas provided for Swing/AWT.

3. World Wind application programming interface applies information or imagery to a globe.

4. It is responsible for providing the timing, scene update, and events. It is also responsible for mapping user actions. The rendering of a model is also controlled by it.

5. It is used to both the aggregates a globe and the layers to apply to this globe. It can used to create even the whole universe. The application which uses this can interact with the model to create the view of a globe. This globe can be of the Earth, Mars, etc.

6. It also controls the view of the model which user gets.

A developer can use this in both the ways, he can use many objects with aobject, and aobject can also be used alone. After doing this, this object is then passed to aobject, and then it shows the globe and its layers. File cache persist all data to the local computer. The file cache manages multiple disk storage locations and is accessible through the World Wind singleton. It provides an interactive interface as well as views of the planet or globe and manages all its layers.

3.16.4. Problem faced in Embedding WWJ into Eclipse

As we know that standard widget toolkit is incompatible with AWT and Eclipse uses the standard widget toolkit, while nature of the WWJ is the Swing/AWT that creates the problem if they both are using together. Additionally, AWT and JOGL are tightly integrated, which makes it more difficult to make a port of the AWT interfaces to standard widget toolkit.

To overcome this problem, the Eclipse foundation developed the standard widget toolkit/AWT Bridge, which makes it comfortable to embed AWT/Swing components into
standard widget toolkit. The bridge it is a very simple application programming interface located in the package. This bridge has been part of standard widget toolkit since version 3.0.

In pushing AWT-based WorldWind 3D globe into an Eclipse application via standard widget toolkit, the standard widget toolkit/AWT Bridge plays an important role as without it, it is not possible to solve this problem.

For integrating the applications to incorporate 3D Earth modelling, the best way is using World Wind. This World Wind is a Java technology component which provides all the features for this kind of work. World Wind is continuously trying to provide the data on the planets which are not provided till yet, data on moons of the planets, different stars, about satellites and for weather also. This data is becoming available all the time. It is also some components which are very useful like animation player, layer manager, user interface helpers and drag and drop functionality. Furthermore, World Wind will include application programming interfaces for scripting extensions and RSS feed support. Very soon more data formats which will be natively supported are part of World Wind.