Software Design and Architecture Report
On
HandyMan Project

Prepared by
Md Mynuddin
ID : ASH1825007M
BSSE, NSTU, IIT

Submitted to
Dipok Chandra Das
Assistant Professor
NSTU, IIT

BSc. In Software Engineering
Institute of Information Technology
Noakhali Science and Technology University

Introduction
This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

Purpose
The purpose of the project HandyMan is to provide service men for every service people need.People can find service providers for any problem and any task as well as the service providers also find tasks as a bonus from online.In short, the purpose of this SRS document is to provide a detailed overview of our software product, its parameters and goals. This document describes the project's target audience and its user interface, hardware and software requirements. It defines how our client, team and audience see the product and its functionality. Nonetheless, it helps any designer and developer to assist in software delivery lifecycle (SDLC) processes.

Project Scope
This Software Architecture Document provides an architectural overview of HandyMan System.
Primarily, the main scope of this project is to develop an application based on smartphone device. As more than 80% users of smartphones are using Android operating based mobile device, so we are targeting to implement our system firstly for Android users. Our proposed the system has 2 modules. This SRS is also aimed at specifying requirements of software to be developed but it can also be applied to assist in the selection of in-house and commercial software products. The standard can be used to create software requirements specifications directly or can be used as a model for defining an organization or project specific standard. It does not identify any specific method, nomenclature or tool for preparing an SRS.

Definitions Acronyms and Abbreviations

This subsection contains definitions of all the terms, acronyms, and abbreviations used in the document. Terms and concepts from the application domain are defined.

Service Provider	Who will receive work request via HandyMan and will provide service
Service Taker	Who will send work request via HandyMan, hire the service provider and will receive service
FAQ	Frequently Asked Questions

References
Essential Software Architecture, Second Edition, Ian Gorton , Springer

Overview
Day by day people's business is increasing. Besides, if our daily-used things like refrigerator, air conditioner, gas stove and other things get wasted or any other problem occurs like those then it becomes quite irritating and hard to carry out those somewhere to repair or to find the correct mechanic to repair. To remove this burden our online platform is here to provide quality service providers for almost everything. It will not only provide service providers to people, but also provide jobs to the service providers.Now-a-days maximum of our population use smartphones and 90% of them are based on Android operating system. So that our project HandyMan will be an android application to reach people widely and easily.

User classes and characteristics

There are 3 types of stakeholders in our "HandyMan" app. Such as :

Service Provider: A service provider is a vendor that provides IT solutions and/or services to end users and organizations. This broad term incorporates all IT businesses that provide products and solutions through services that are on-demand, pay per use or a hybrid delivery model. A service provider's delivery model generally differs from conventional IT product manufacturers or developers. Typically, a service provider does not require purchase of an IT product by a user or organization. Rather, a service provider builds, operates and manages these IT products, which are bundled and delivered as a service/solution. In turn, a customer accesses this type of solution from a service provider via several different sourcing models, such as a monthly or annual subscription fee.

Service Taker: Services have subscription payment models rather than point-of-sale models. For example, HandyMan is a service provider, not a movie retailer. Software as a service is a more lucrative model than selling a single unlimited licence. For example, Microsoft makes less money selling you a copy of Office than selling you a subscription to Office 365, which is a service. The benefits to the consumer include continuous improvement of the software over time. Google offers App Engine which is a platform as a service. The platform can run your service with features like scaling to support any load of traffic. In this model they are selling access to their infrastructure as a service. That kind of service cannot easily be packaged and sold as a product. They tried selling hardware at one point to encapsulate their search technology as a physical product. I don't think it saw widespread adoption and they stopped doing that. I think they learned that services are easier to maintain and make more money.

bKash/DBBL Organization: bKash is a mobile financial service in Bangladesh operating under the authority of Bangladesh Bank as a subsidiary of BRAC Bank Limited. This mobile money system started as a joint venture between BRAC Bank Limited, Bangladesh and Money in Motion LLC, United States of America. bKash users can deposit money into their mobile accounts and then access a range of services, in particular transferring and receiving money domestically, making payments and can recharge prepaid mobiles easily. We use bKash in our app and also added DBBL.

Architectural Representation

This document presents the architecture as a series of views; use case view, logical view, process view and deployment view. There is no separate implementation view described in this document. These are views on an underlying Unified Modeling Language (UML) model developed using Rational Rose.

Use-Case View

A description of the use-case view of the software architecture. The Use Case View is important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

The HandyMan Use cases are:

- ➔ **Set Location :** In the opening of HandyMan, service taker may need to set location. It would help to find the near location of the service provider, besides service taker can find his location. Our system will provide him/her a near location available service provider
- ➔ **Search Service :** Service taker can search any type of service what he/she looking for
- ➔ **Show Service list :** After searching, Service taker sees a multiple Service list. He or she can choose a service or multiple service
- ➔ **See Provider List :** Service taker chooses his service. Then the system shows Some of his service related service provider names. Service taker can see any of them profile.
- ➔ **Give Rating :** Service taker can give rating to service provider who has done his/her service. He /She can give ratings 1 to 5 star
- ➔ **Send "WorkRequest" :** Service taker can see any service provider profile so he /she choose service provider for his/her service and sends "work request"
- ➔ **Cancel "WorkRequest" :** After sending "work request", the service provider does not respond or the service taker does not want the service at the moment. So, he/she can cancel the "work request".
- ➔ **Pay Money :** After the service provider accepts the request, the service taker pays money for his/her service using bkash/DBBLs.
- ➔ **Update Profile :** Service taker has own profile. He/she can change his/her information like name, email, Mobile no etc

- **Change Activities Status :** Service provider can his/her Active status using turn on or off.
- **Cancel request :** Service provider can cancel the service taker's "work request if he/she doesn't like it.
- **Accept request :** Service taker sends work request to Service provider. Service Provider can accept the request if she/he wants.

Architecturally-Significant Use Cases

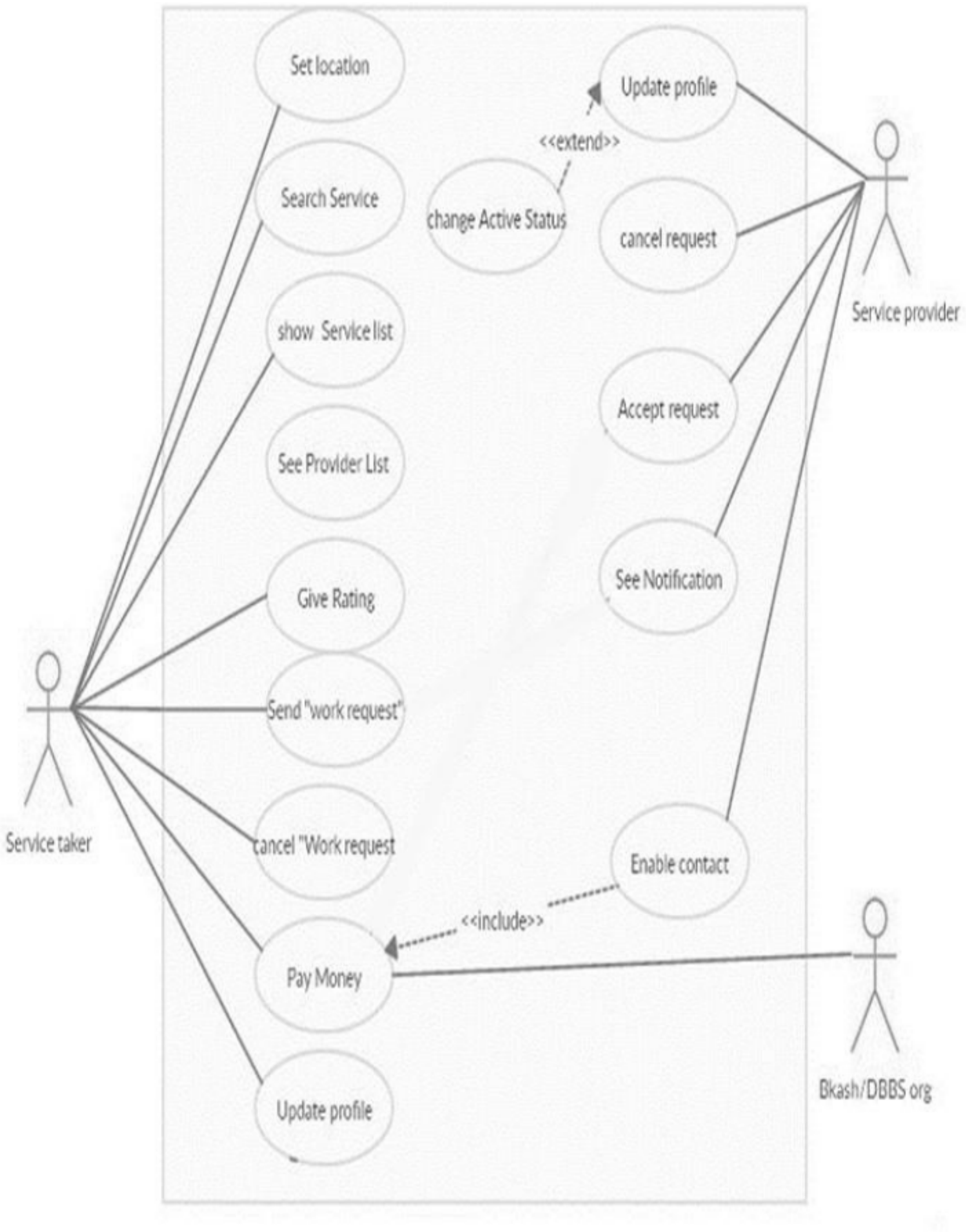


Diagram Name : Architecturally Significant Use-Case

Logical View

A description of the logical view of the architecture. Describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers. Also describes the most important use-case realizations, for example, the dynamic aspects of the architecture. Class diagrams may be included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.

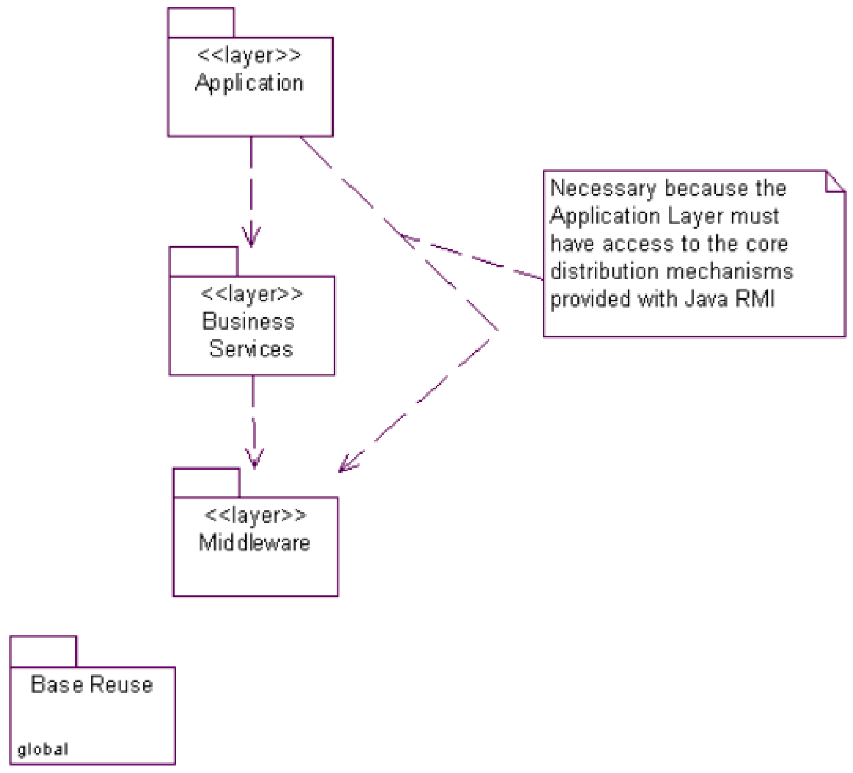
The logical view of the course HandyMan system consists of the 3 main packages: User Interface, Business Services, and Business Objects.

The User Interface Package contains classes for each of the forms that the actors use to communicate with the System. Boundary classes exist to support set location, Search Services, Show Service List, See Service provider list, Give Ratings, Send WorkRequest, Cancel WorkRequest , Pay Money , Update Profile and Pay Money.

The Business Services Package contains control classes for interfacing with the billing system, controlling Contract, and managing the request like accept and cancel request.

The Business Objects Package includes entity classes for the HandyMan artefacts (i.e. Service provided based on location) and boundary classes for the interface with the Service Provide System.

Architecture Overview – Package and Subsystem Layering



<<layer>>
Application
<<layer>>
Business
Services
<<layer>>
Middleware
Base Reuse
global
Necessary because the Application Layer must have access to the core distribution mechanisms provided with Java RMI

Application layer : This application layer has all the boundary classes that represent the application screens that the user sees. This layer depends upon the Process Objects layer; that straddles the separation of the client from mid-tier.

Business Services layer : The Business Services process layer has all the controller classes that represent the use case managers that drive the application behavior. This layer represents the client-to-mid-tier border. The Business Services layer depends upon the Process Objects layer; that straddles the separation of the client from mid-tier.

Middleware layer : The Middleware layer supports access to Relational DBMS and OODBMS.

Base Reuse : The Base Reuse package includes classes to support list functions and patterns.

Process View

A description of the process view of the architecture. Describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations. Also describes the allocation of objects and classes to tasks.
The Process Model illustrates the HandyMan Services organized as executable processes. Processes exist to support set location, Search Services, Show Service List, See Service provider list, Give Ratings, Send WorkRequest, Cancel WorkRequest , Pay Money , Update Profile and Pay Money.

Simple Process :

1.User can see available service based on their location
2.User choose services and send workrequest or can cancel request.
3.If choose a service, the service provider can accept or reject the request.
4.If accept the request and complete his / her job , the service taker will pay money to the service provider.

Deployment View

A description of the deployment view of the architecture Describes the various physical nodes for the most typical platform configurations. Also describes the allocation of tasks (from the Process View) to the physical nodes.
This section is organized by physical network configuration; each such configuration is illustrated by a deployment diagram, followed by a mapping of processes to each processor.

Service taker =>Service Provide Server (Need Internet access)
Service Provider => Service Provide Server (Need Internet access)
Services => Services Provide Server (Need Internet access)
Pay Money or Billing System => Service Provide Server (Need Internet access)

Size and Performance

The chosen software architecture supports the key sizing and timing requirements, as stipulated in the Supplementary Specification

1. The system shall support up to 2000 simultaneous users against the central database at any given time, and up to 500 simultaneous users against the local servers at any one time.
2. The system shall provide access to the available services database with no more than a 10 second latency.
3. The system must be able to complete 80% of all transactions within 2 minutes.
4. The client portion shall require less than 20 MB disk space and 32 MB RAM.

The selected architecture supports the sizing and timing requirements through the implementation of a client-server architecture. The client portion is implemented on local campus PCs or remote dial up PCs. The components have been designed to ensure that minimal disk and memory requirements are needed on the PC client portion.

Quality
The software architecture supports the quality requirements, as stipulated in the Supplementary Specification.

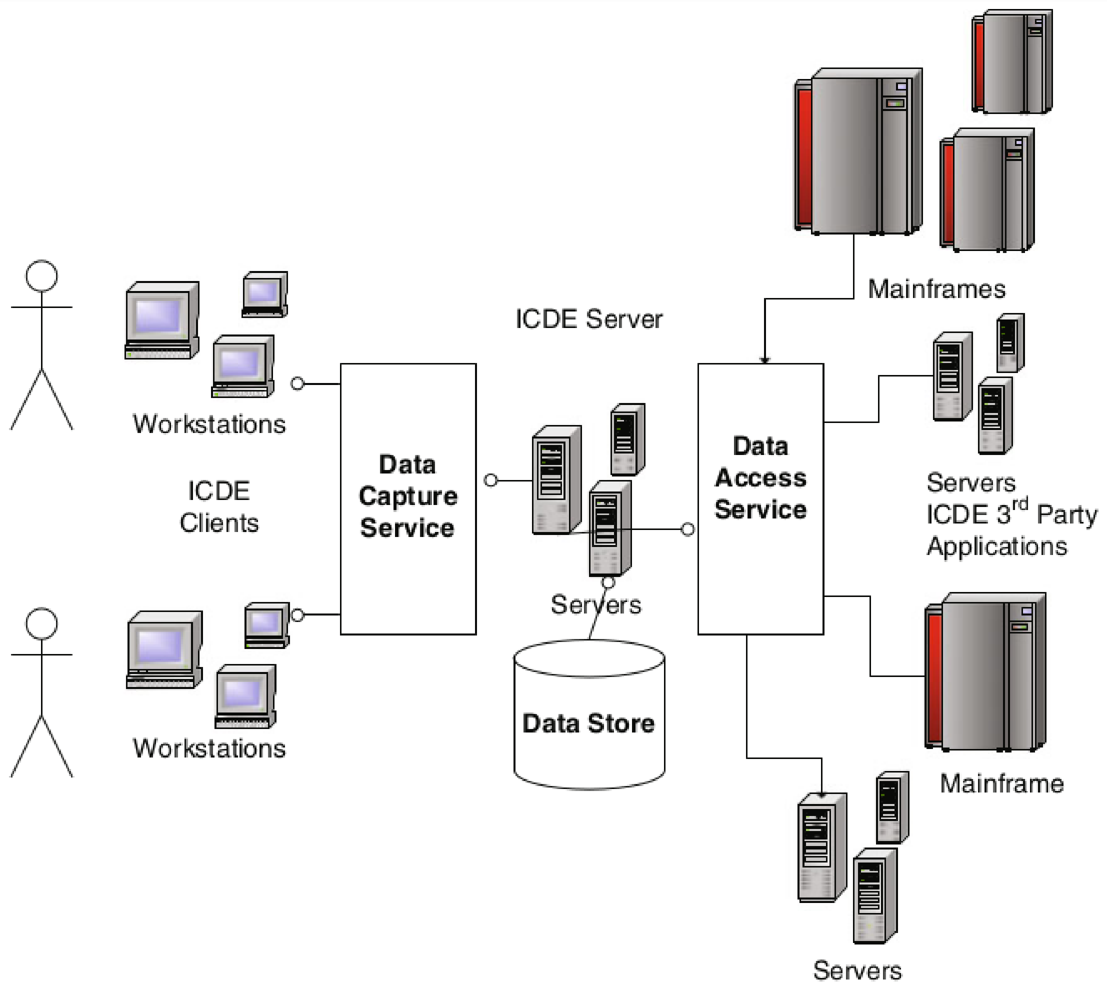
1. The android user-interface shall be Windows 95/98 compliant.
2. The user interface of the HandyMan shall be designed for ease-of-use and shall be appropriate for a ANdroid-literate user community with no additional training on the System.
3. Each feature of the HandyMan shall have built-in online help for the user. Online Help shall include step by step instructions on using the System. Online Help shall include definitions for terms and acronyms.
4. The HandyMan System shall be available 24 hours a day, 7 days a week. There shall be no more than 4% down time.
5. Mean Time Between Failures shall exceed 300 hours.
6. Upgrades to the Android client portion of HandyMan shall be downloadable from the UNIX Server over the internet. This feature enables students to have easy access to system upgrades

The ICDE System
The Information Capture and Dissemination Environment (ICDE) is part of a suite of software systems for providing intelligent assistance to professionals such as financial analysts, scientific researchers and intelligence analysts. To this end, ICDE automatically captures and stores data that records a range of actions performed by a user when operating a workstation.

For example, when a user performs a Google search, the ICDE system will transparently store in a database:

- ➔ The search query string
- ➔ Copies of the web pages returned by Google that the user displays in their browser

This data can be subsequently retrieved from the ICDE database and used by third-party software tools that attempt to offer intelligent help to the user. These tools might interpret a sequence of user inputs, and try to find additional information to help the user with their current task. Other tools may crawl the links in the returned search results that the user does not click on, attempting to find potentially useful details that the user overlooks.

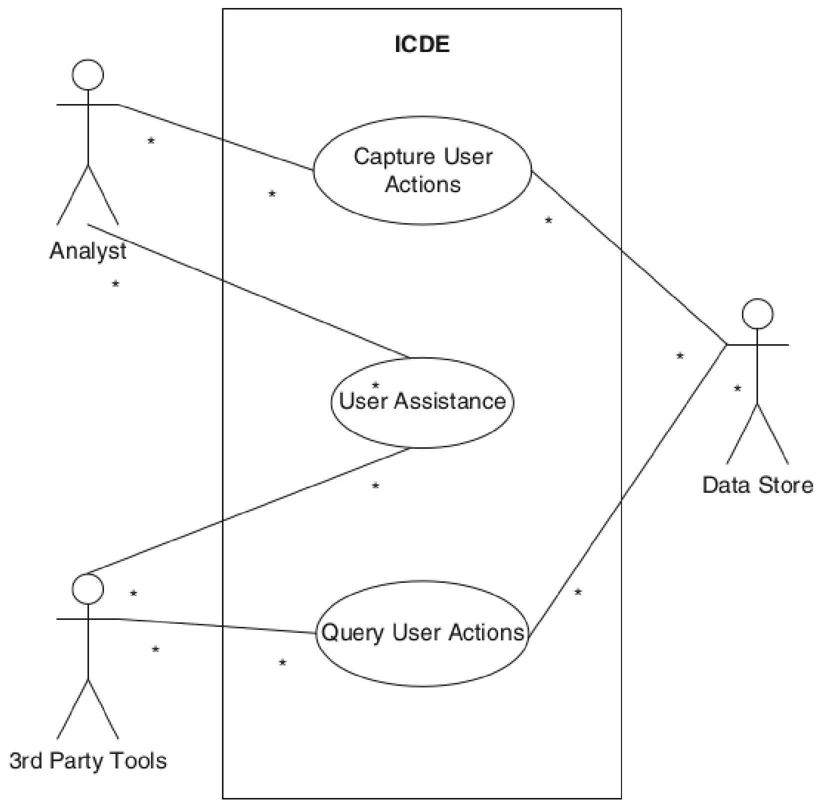


ICDE Context Diagram

A use case diagram for the ICDE system is shown in ICDE context diagram The three major use cases incorporate the capture of user actions, the querying of data from the data store, and the interaction of the third party tools with the user.

ICDE Use cases

- Capture user action
- User assistance
- Query user action



ICDE use cases

Our HandyMan System is not providing intelligent assistance to professionals such as financial analysts, scientific researchers and intelligence analysts. According to the ICDE definition and details understanding i think our HandyMan system is not compatible with ICDE System.