Con-Do-It: An Prototype for Plumber Aggregration Platform

Sujit Kumar Chakrabarti, Amar Pratap Singh, Tejdeep Gutta, Pankaj Jadhav

December 24, 2022

Part I

Preliminaries

Chapter 1

Introduction

1.1 Plumber Aggregation Platform

A plumber aggregation platform is a web/mobile platform envisaged by APPL. Some of the important features of the platform are as follows:

- Customers connect to plumbers who are located close to them and avail their services.
- Complaint management and processing
- Payment processing
- Plumber management
- Inventory management

1.2 Con-Do-It

Con-Do-It is a prototype for the plumber aggregation platform. It is meant to help elicit the requirements of an actual product of a similar description that would be developed by APPL (likely in partnership with a professional software development firm). Con-Do-It aims to aid requirement elicitation and design of the aggregrator platform. Its explicitly excludes, as one of its goals, evolving into an actual product as development of such a product will require skills and resources of a professional software development company, both in terms of development and operations.

1.3 This Document

This document reports all the work that has been done in IIITB in the period August 2022 through December 2022 in the development of the Con-Do-It prototype. This includes requirement modelling, design, user instructions, test design along with thoughts on the further roadmap to the development of the plumber aggregator platform.

The goal of this document – along with the prototype – is to act as an introductory reference in the initial stages of the development of the Plumber Aggregator Platform. We have tried our best to push the level of details as close as possible as would be sufficient to develop a full-fledged application. However, it actually is not quite at that level. The business and deployment conditions in real-time are likely to be very dynamic and impossible to predict that this stage. Hence, beyond a particular level of detail, our models may drift away from the reality. We believe that further details should be worked out as a part of an agile software development process for the aggregator platform.

1.3.1 Intended Audience

This document is written for the benefit of software development team(s) who undertake the final development of the Plumber Aggregator Platform.

Chapter 2

Plan

2.1 Project Deliverables

The project deliverables are as follows:

1. A list of requirement features, their elaboration as use cases, scenarios and other appropriate forms as appropriate
2. Analysis of requirements in terms of their relative importance/priority, interaction and dependence
3. Analysis of non-functional requirements
4. Working prototype implemented and piloted in consultation with APPL.

2.2 Key Features

1. Location and rating based plumber allocation algorithm
2. Call Tracking and Status update
3. Rating mechanism for Plumber and the customer and other algorithms that are required for the solution
4. Pricing, Invoicing process and payment option
5. Plumber reimbursements
6. Knowledge bank (Link to technical contents)
7. Gamification to keep users engaged on the platform
8. Omni Channel capability for Consumer to raise a support request. etc

2.3 Plan

PI will devote approximately half a day (3 hours) of work every week on the project. This will include an hour of discussion with the project students. Two students of 3rd Year iMTech will devote approximately one day per person of work to the project. This will involve software development, team interaction, documentation, research and any other project related activities as necessary.

The team will meet at least once every week for update and weekly planning and technical discussion. APPL personnel will join these meetings every alternate week for brainstorming and feedback.

2.3.1 Timeline

An iterative prototype driven approach will be followed to execute the project. The deliverables and their expected timelines are as follows:

Deliverable	Date
Initialisation	
Requirement features List	August 29, 2022
Iteration 1	
Requirement analysis (v1)	September 05, 2022
Prototype design (storyboard)	September 12, 2022
Prototype implementation (v1)	September 26, 2022
Pilot (v1)	October 03, 2022
Iteration 2	
Feedback and requirement analysis	October 10, 2022
Prototype design and implementation	October 24, 2022
Iteration 3	
Feedback and requirement analysis	October 31, 2022
Prototype design and implementation	November 7, 2022
Ramp down	
Requirement Analysis Document	November 14, 2022
Architectural Design Document	November 21, 2022
Buffer	

Part II

Requirement Analysis

Chapter 3

Requirements

3.1 Stakeholders

1. APPL
2. User: Customer
3. User: Plumber
4. User: Service manager/officer
5. User: Super user
6. Suppliers

The various user roles/categories are designated different privilege levels in ascending order as follows:

1. Level 1: Customer and plumber
2. Level 2: Service manager
3. Level 3: Super user

A user belonging to a higher privilege level is allowed to invite any individual to sign up at a lower privilege level. Likewise, user belonging to a higher privilege level is allowed to delete any account at a lower privilege level if required. More details about the sign up features are provided in Section 3.6.1.

3.2 User

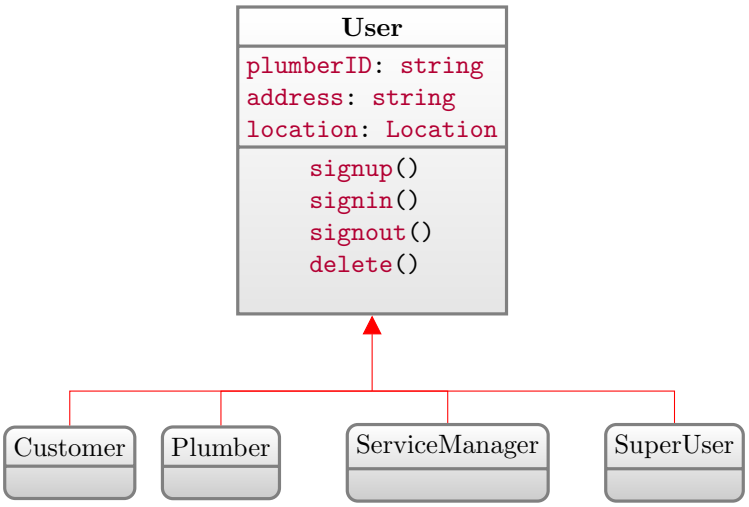


Figure 3.1: User class

3.3 Customer

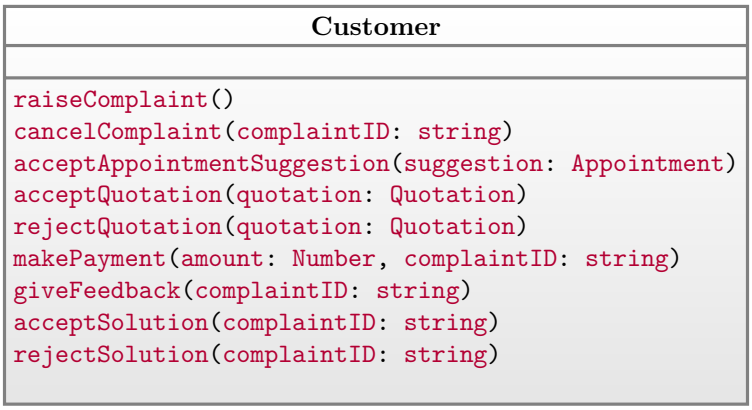
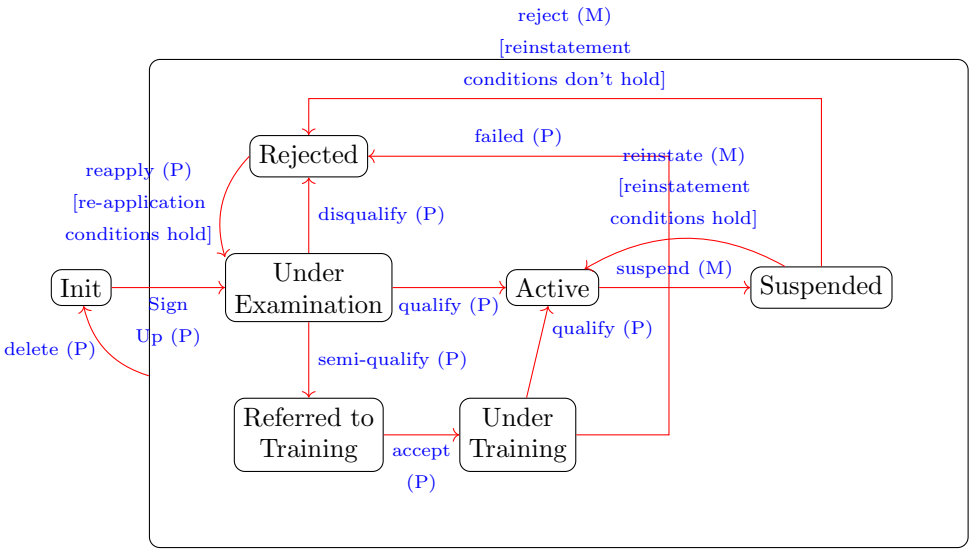


Figure 3.2: Customer class

A customer's primary role is to raise complaints as required. At various stages of processing a complaint, a customer would need to perform various actions like:

- select a plumber as suggested by the system or ask for more suggestions
- select a time slot as suggested by the system or ask for more suggestions
- approve or reject a quotation as raised by a plumber
- give feedback on the processing of a complaint
- make necessary payments

Sign-up and sign-in are necessary prerequisite use cases for performing any other actions.



P	Plumber
M	Manager

Figure 3.4: Plumber States

3.4 Plumber

Plumber
`specialisation: Set<ServiceType>`

`acceptAssignment(complaintID: string)`
`abandonComplaint(complaintID: string)`
`raiseQuotation(quotation: Quotation)`
`fillJobCard(jobcard: JobCard)`

Figure 3.3: Plumber class

A plumber deals with installation and repair requests originating from customers or otherwise. A plumber would be able to view all complaints assigned to him so far. The plumber can respond to active complaints assigned to him. He can order spares from the company warehouse. He can communicate with the customer and seek his approval to use any spares required for a repair job. He would be allowed to check out spares only if the customer has consented to pay for the spares/components.

A plumber must sign up as a plumber before he can avail the features of the applications. While signing up, a plumber mentions his specialties on the system along with other details. Once signed up, the plumber would go through an examination/verification. If found suitable, the plumber would be inducted to the fleet of plumbers by the service manager. Otherwise, the plumber may either be sent for an upskilling training programme or may be rejected. There is a possibility of temporarily suspending a plumber due any organisational decision, or of removing/dismissing a plumber. A dismissed plumber will be allowed to re-apply on meeting certain conditions as defined by the organisation.

3.4.1 Scenarios of Complaint Processing

1. Plumber sees a new complaint assigned to him.
2. Accepts

3. Customer gets notified about the plumber details.
4. System looks at the earliest available slots with the given plumber and shares with a list of suggested visit slots with the customer.
5. Customer selects a visit slot from among the provided slots. In case the customer doesn't find any of the suggested slots suitable, the system has to suggest more slots. For example, if the job is deemed urgent by the customer then later slots may not work for him. In such cases, other plumbers should be considered. On the other hand, if the customer is keen to be serviced by a particular plumber, and for that, he doesn't mind a later slot, such will be the suggestions made by the system.

At the time of raising a complaint, the customer may be asked to provide some helpful inputs, e.g.:

- Preferred times
- Plumber preference
- Urgency
- ...

The need to provide these details should be optional. It should be possible for the customer to go ahead with the complaint raising with/without providing any piece of information that the system solicits. The system algorithmically does the plumber and time allocation based on the information provided by the customer.

There are two types of services provided:

1. Installation service
2. Repair and maintenance

Installation service involves components and procedures that are predictable. Hence, a quotation and payment can be done right up front at the time of raising the complaint. However, repair and maintenance work may incur unforeseeable costs. Hence, the final charges of a repair/maintenance work can be determined only after the completion of the task.

Repair and maintenance can be done for two types of customers: existing customers and outside customers. Pricing will be different for these two categories. Existing customers (who are availing service for equipments under warranty) will not be charged for visit.

For outside customers, visiting charges will be mandatory.

If the spare parts costs are deemed too high by the customer, and he wants not to go ahead with the service, the ticket will be closed as satisfactory.

No wallet option provided.

Charges will be decided based on various rules (e.g. location etc.) for any service/component. The service department will have an interface which will be used to define these rates.

3.5 Use Cases

1. Raising of plumbing complaint
2. Assignment of plumber to a complaint
3. Tracking of complaint
4. Closing of complaint
5. Feedback on service
6. Re-assignment of complaint

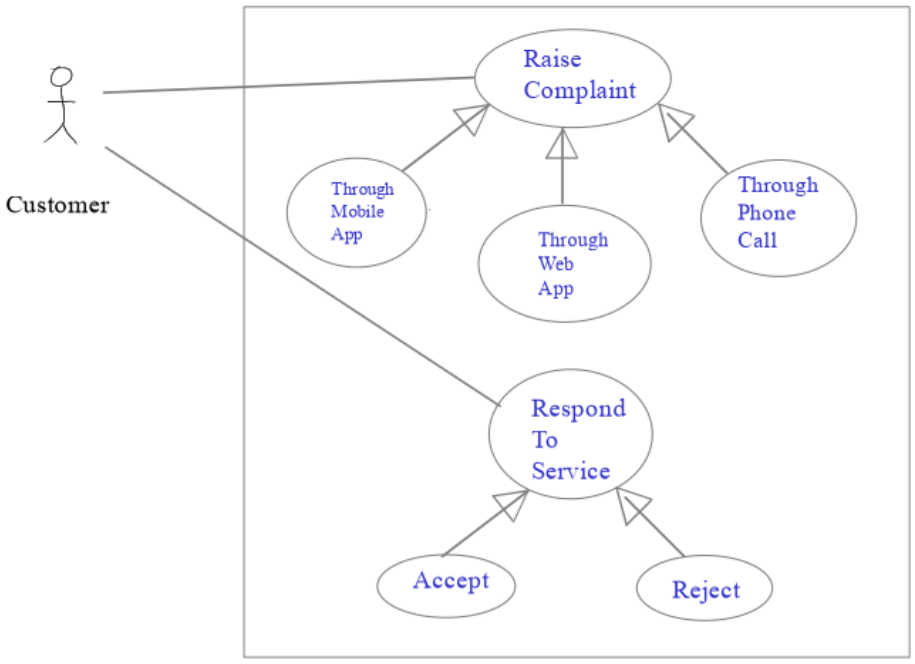


Figure 3.5: Use cases: Customer

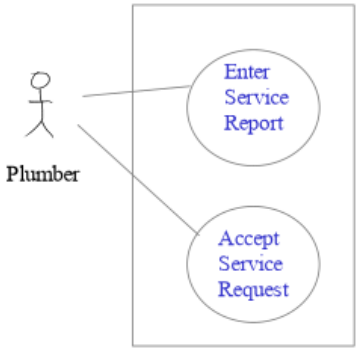


Figure 3.6: Use cases: Plumber

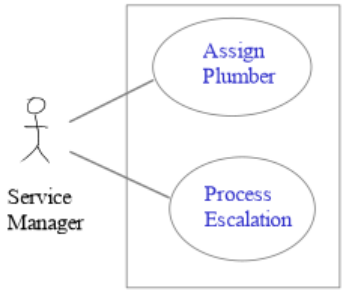


Figure 3.7: Use cases: Service manager

3.6 User Accounts

3.6.1 Sign Up

Any user (except super user) would have to sign up to become part of the system. At the time of sign up, a user would need to choose a role (e.g. customer, plumber or service manager). The details of the sign up process will depend on the role chosen by the user.

Common Sign up Features

Personal details like email ID and phone number will be collected at the time of sign up. It is desirable that for any given role, an individual should hold at most one account. Hence, the system will mandate that, for a particular role, the personal details should be distinct. However, it is fine for the same individual to hold multiple accounts with maximum one account in each user category (customer, plumber and service manager). In extreme cases, the organisation may decide to remove an account in case of abuse, default or any other issue.

Customer Sign Up

A customer can sign up directly without any verification.

Plumber Sign Up

A plumber can sign up directly without any verification. However, to become active, he would need to undergo certain verification and/or training based on his current skill level and what is expected in the role he is seeking.

3.6.2 Sign In

A user can sign in/login using his/her credentials. Various options can be provided for sign in, e.g., password based or OTP based or Google/Facebook based or multilevel signin (e.g. Online SBI). If the user forgets his/her credentials, the system will provide appropriate help to reset the password etc.

At the time of sign in the user must mention which user role he/she wishes to sign in as. For the sign in to succeed, the user must have an account in that role. If the user wishes to change his role, he would have to logout and login again with a different role.

3.6.3 Account Deletion

Due Clearance

What do we do when a person with present dues tries to delete his account?

3.7 Complaint

The states of a complaint object may be described in more detail through additional states.

- **Initial:** The initial state. This means that the complaint in question hasn't yet been raised.
- **Raised:** This is the state of the complaint immediately after it has been raised.
- **Assigned:** A raised complaint is assigned to a plumber, either manually by the service or algorithmically by the system, taking it to *Assigned* state.
- **Accepted:** Once a plumber accepts a complaint

- **Under examination:** When a plumber visits the customer site, his first action is to report the visit. This marks the examination to be ON, and the complaint moves to *under examination* state.
- **Quotation raised:** After examination, the plumber examines the issue and generates an estimate or quotation of the approximate expenditure. This needs to be approved by the customer before the service can proceed further. At this point, the complaint in in *quotation raised* state.
- **Under execution:** Once the customer approves the quotation, the plumber starts working on the issue. The complaint is *under execution*.
- **Task completed:** Once the task is completed to the satisfaction of the plumber, he marks it as *completed*. The customer's feedback + acceptance is solicited by the system. If the customer finds the service not up to his/her satisfaction he/she may *reject* it. This takes the issue back to the *raised* state. Alternatively, the customer may go ahead and *accept/approve* the solution. This takes the complaint to *payment pending* state.
- **Payment pending:** The state the complaint is in when a task has been marked completed by the plumber and the same has been found acceptable by the customer.
- **Payment completed:** In payment pending state, the customer is required to pay up the expenses for the service. Once this is done, the complaint movesto the *payment completed* state.
- **Done:** Once payment is done, the complaint reaches its logical conclusion. This is the *done* state.
- **Abandoned:** When the quotation is raised, the customer may find it unaffordable and reject it. This will take the complaint to the *abandoned* state.
- **Cancelled:** Before the plumber starts executing a particular task, the customer has the freedom to cancel the complaint. If so done, the complaint reaches a *cancelled* state.

These states and their relation with each other have been presented in Fig. 3.8.

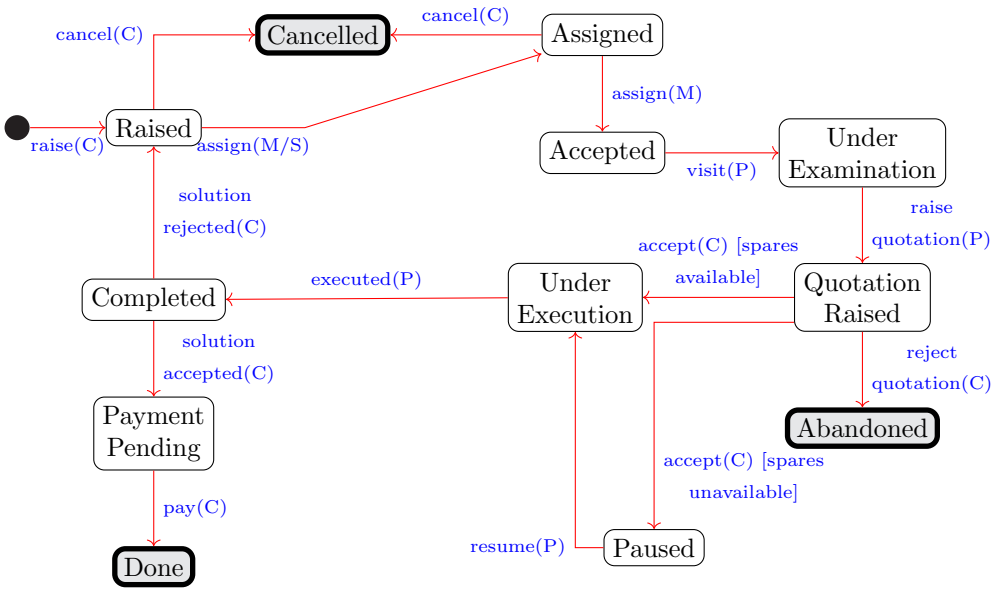


Figure 3.8: Complaint: State machine

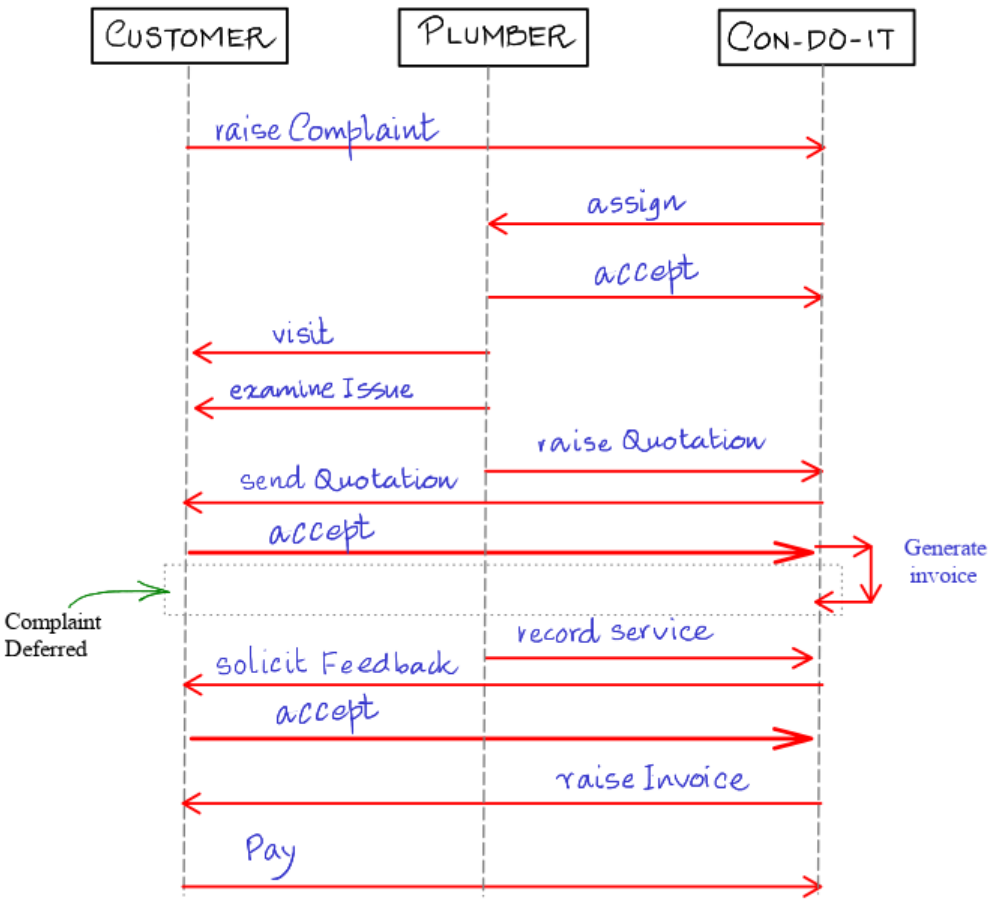


Figure 3.9: Complaint resolution: Basic

Solution Rejection and Pending Amount

We need to work out in more detail, what needs to be done when a customer rejects a solution that has already incurred some expenditure. In some cases, the incurred expenses should carry over straight to the next iteration of the solution. In certain cases, when the solution is proved to have altogether failed, the expenses should be on the company and shouldn't be forwarded to the next iteration. In some cases, a partial amount may get waived. All these are matters of discretion requiring human intervention.

3.8 Complaint Resolution

3.9 Component/Spare Part Procurement

The taxonomy of purchase mechanisms is presented pictorially in Figure 3.11. The ovals enclosed within dotted boxes are those for which the Con-do-it application must have additional features. Customer purchase is beyond the purview of the application.

- **Components available:** If a spare part/component required for a specific service is not available immediately, it would have to be procured. There are a number of ways in which a component may be procured.
- **Available in company warehouse/store:** In this condition, the plumber can order these from the company stores after taking approval from the customer. In such case, the corresponding cost will automatically get added to the invoice.

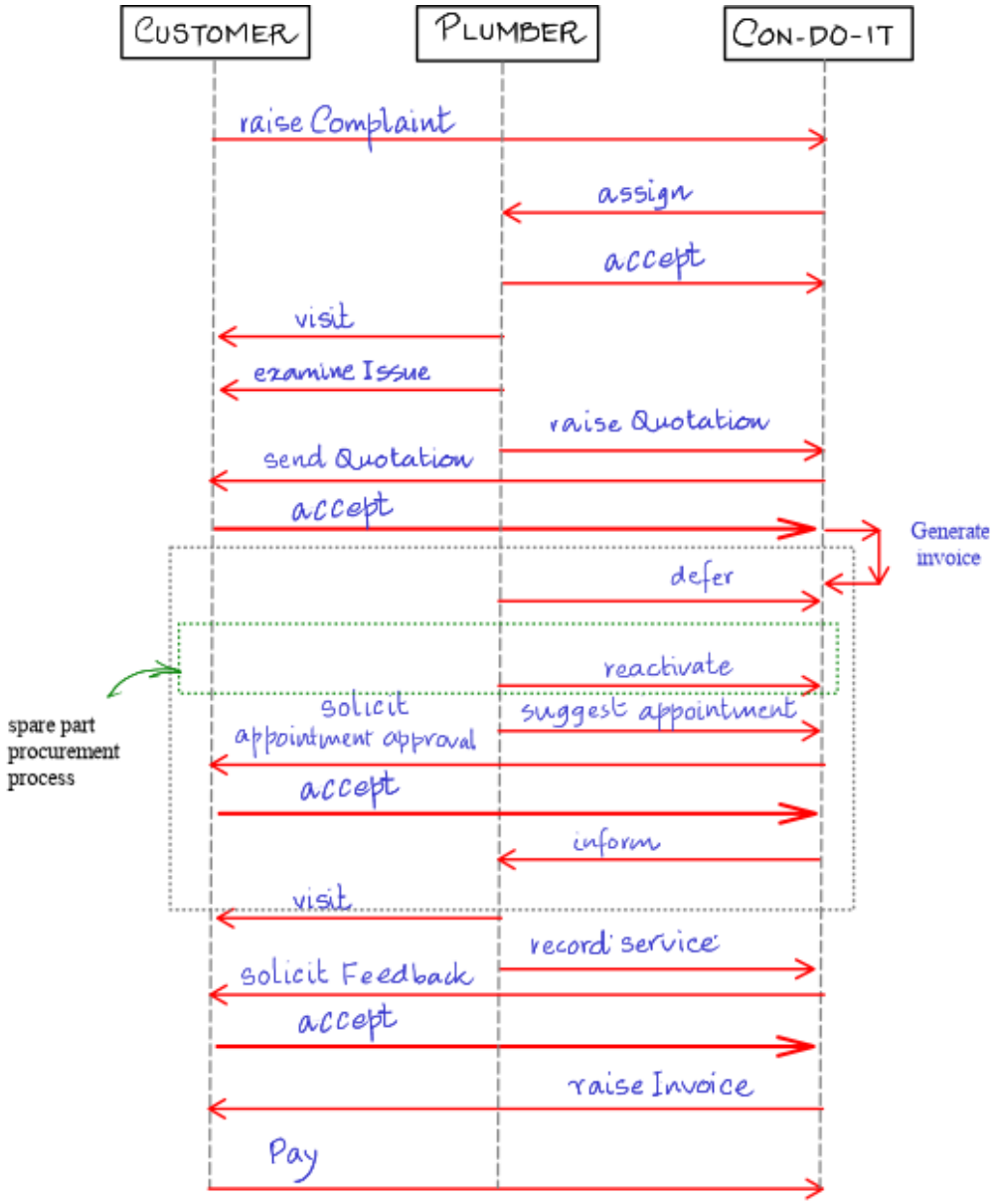


Figure 3.10: Complaint resolution: Component Procurement

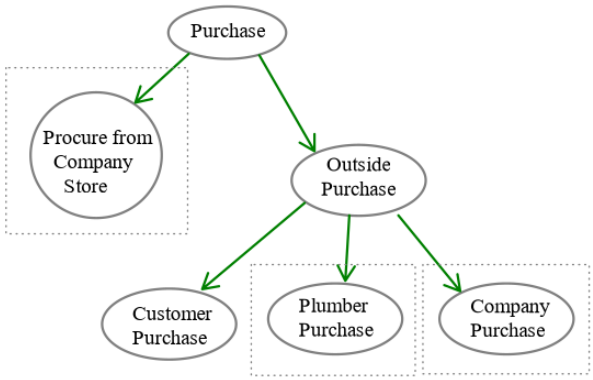


Figure 3.11: Component Procurement

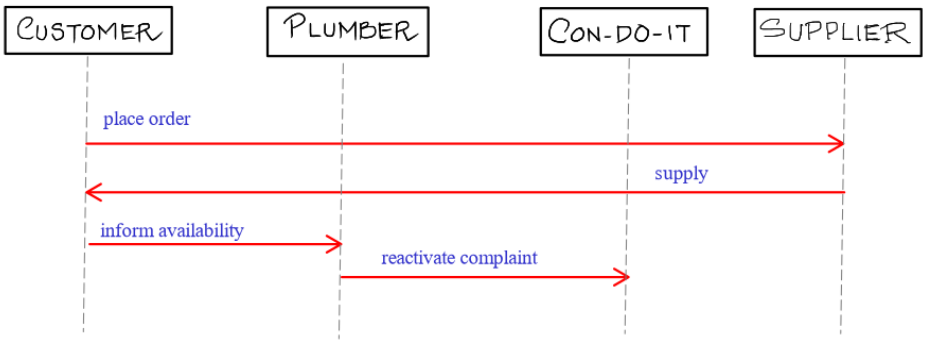


Figure 3.12: Component Procurement: Customer Purchase

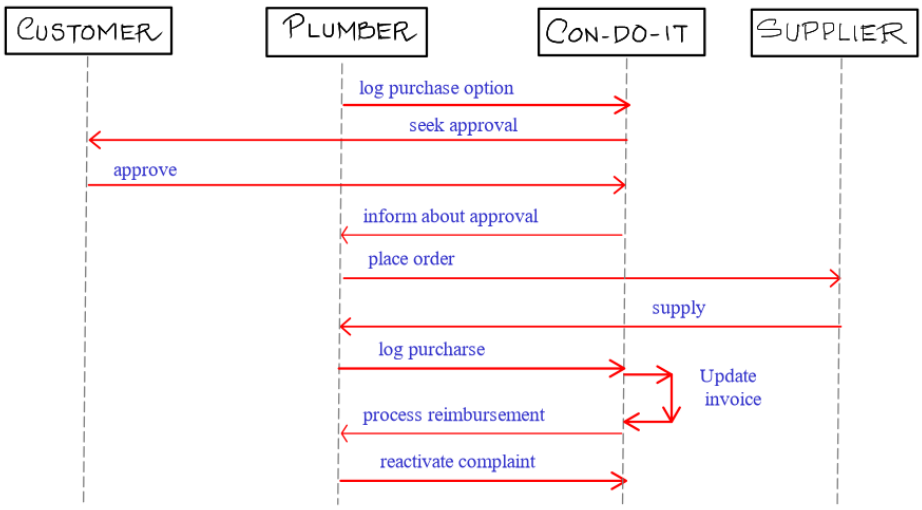


Figure 3.13: Component Procurement: Plumber Purchase

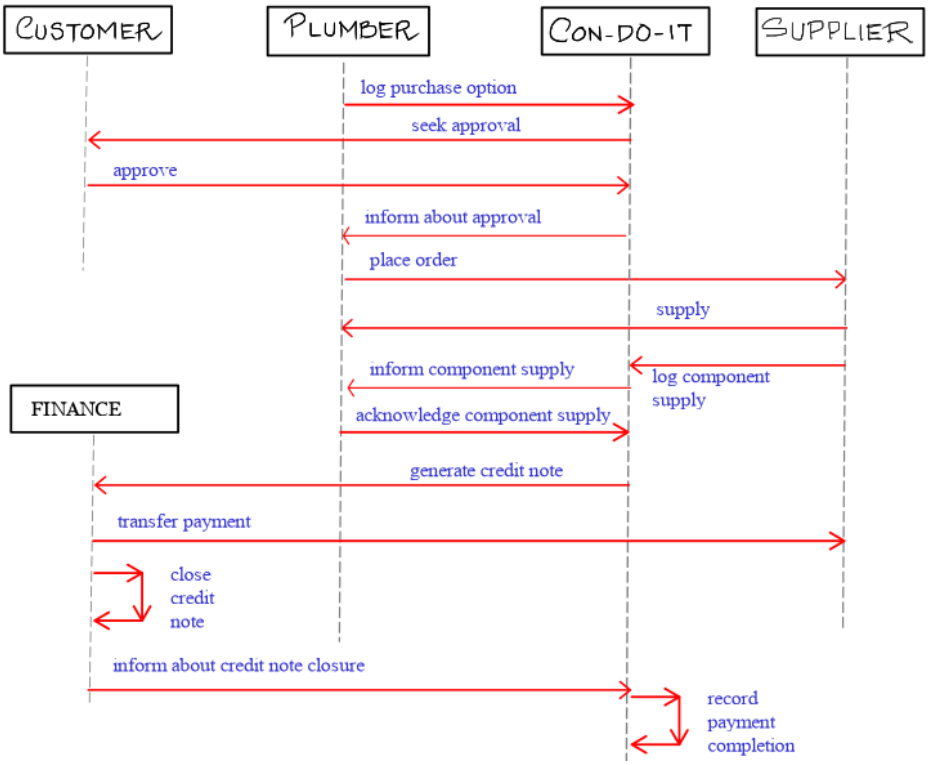


Figure 3.14: Component Procurement: Plumber Purchase (Cashless)

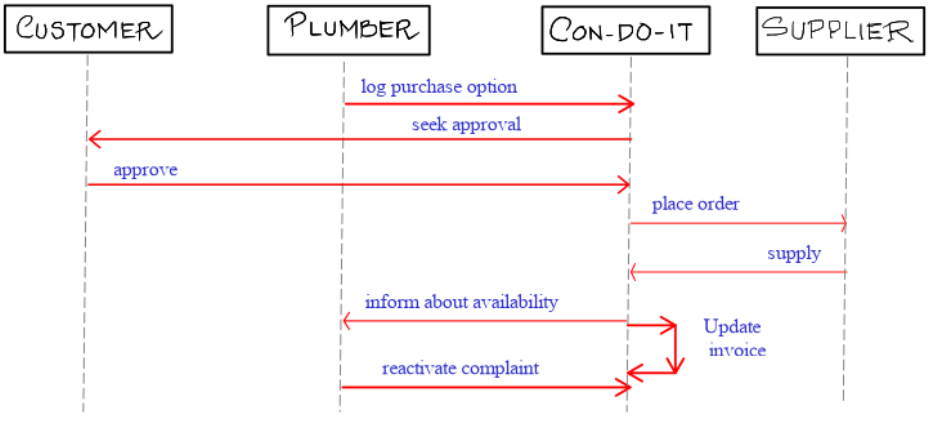


Figure 3.15: Component Procurement: Company Purchase

- **Outside purchase:** In this case, the spare part may have to be procured from some other source. Here, the customer may purchase it from outside and provide the same to the plumber. No changes need to be done to the invoice. The other option is that the plumber purchases it on his own, and raises a reimbursement request to the company. The company, after ensuring that necessary approval from the customer has been obtained, releases the money. Another alternative could be to purchase the part through the company. In this case, the company verifies that necessary approval from the customer has been obtained, and then orders the part on behalf of the plumber.

QUESTION (Sujit)

Suppose a plumber chooses to procure a spare from outside supplier through the company. I assume, this would happen through the purchase department (or something similar). Similarly, if the plumber procures a spare from outside and requests reimbursement, the finance department would have to reimburse the same. Hence, the both the above cases, the system would need to interface with some other internal system, viz. purchase requisition system in the former case and finance system in the latter.

Is my understanding correct?

ANSWER (Pankaj Jadhav)

There are two scenarios in Product or Spare requirement:

Free of Cost (FOC): Plumber will seek approval from APPL Service manager by submitting the defect and purchase proof image > Plumber pick the product from nearest dealer or DP (Distributor Partner) on loan basis after approval > Defective Product or Spare will be submitted to APPL Service manager by plumber along with defect return challan generated from the system against the complaint > APPL Service manager will route the challan to APPL finance for Credit note[1] > Finance will pay the amount to DP and mention the CN closure note in CRM.

Billing or Chargeable: Plumber will generate the quote from system or CRM along with the nearest dealer or DP name (Rate are predefined and fixed) > Get the quote approved from end customer > Reach out to nearest Dealer or DP for getting the necessary Product or Spare along with proper tax invoice > Collect the part charges from customer after replacing it > Submit the payment to Dealer or DP from where he/she took the material on loan basis. This process is applicable for non APPL brand product however the plumber will be incentivized only on sale of APPL product or spare against uploading of valid DP or dealer invoice.

Part III

Design

Chapter 4

Architecture

4.1 Context

Future Work on Architectural Design
Work on the precise definition of interfaces between sub-systems and their deployment models.

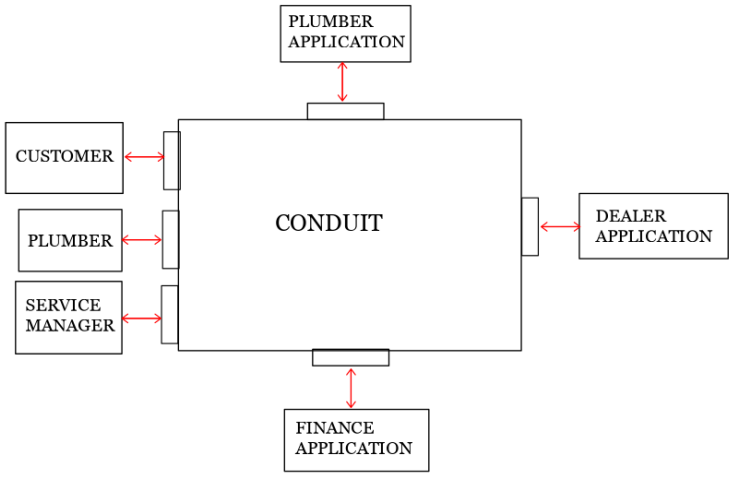


Figure 4.1: Architectural Context

Chapter 5

User Interface

5.1 Common

5.2 Customer Interface

5.3 Plumber Interface

The plumber's dashboard's most important item is the "My Complaints". When the plumber selects this option, the list of complaints will be shown to the plumber. The complaints may be sorted as per different criteria, e.g. name of the customer, date of raising, date of acceptance, date of completion, state (e.g. raised, completed etc.)

When the plumber selects a particular complaint, what gets shown to him depends on the state of the complaint. For example, if a complaint has already been accepted, there should be no option provided to accept/reject it. If a complaint has been paused, the plumber should not be allowed to do anything in it unless it is resumed.

This restriction of available choices will limit the number of decisions or choices the plumber has to make at any point in time. This will lead to increased usability.

5.4 Service Manager Interface

5.4.1 Plumber Suggestion

A service manager will be responsible for many discretionary activities. For now, many of the processes which will end up as automated will be routed through the service manager. A service manager would have the provision to either explicitly delicate the process to an algorithm, or execute it himself. For example, when a new complaint is raised, depending on the region, the same will be routed through a regional service manager. The service manager may then decide to manually produce a set of suggestions for the customer or invoke the algorithm to do it automatically.

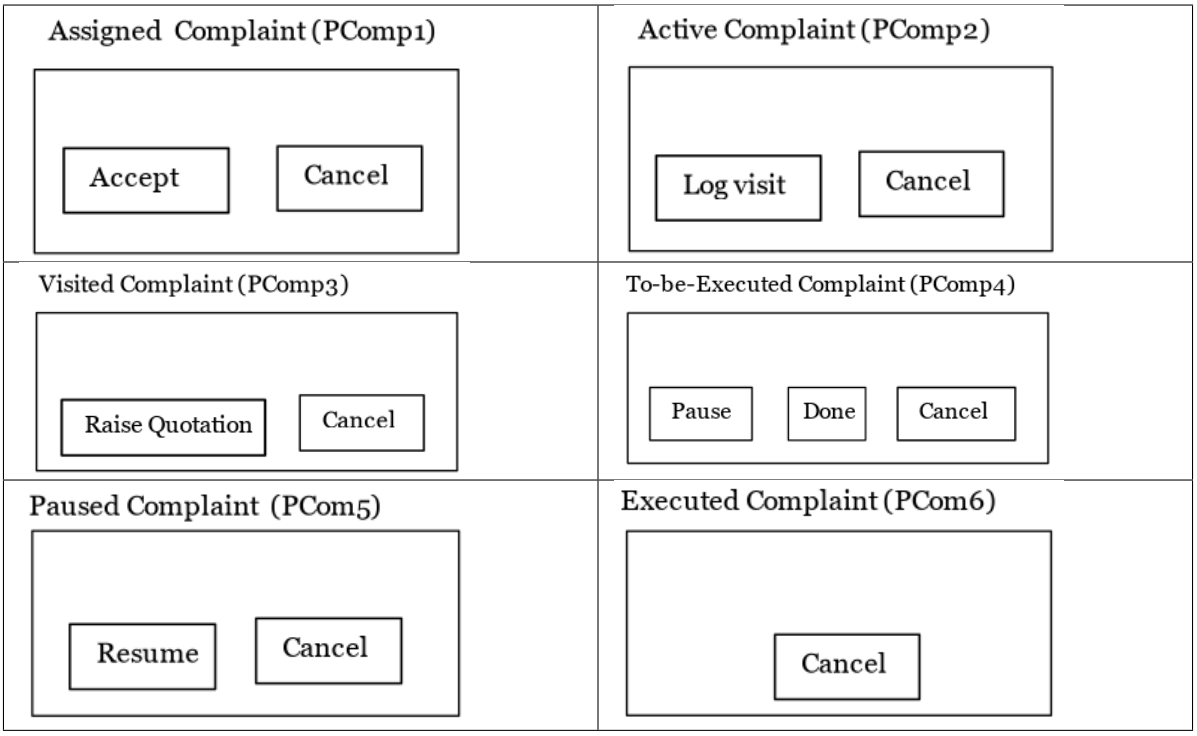


Figure 5.1: Plumber screens

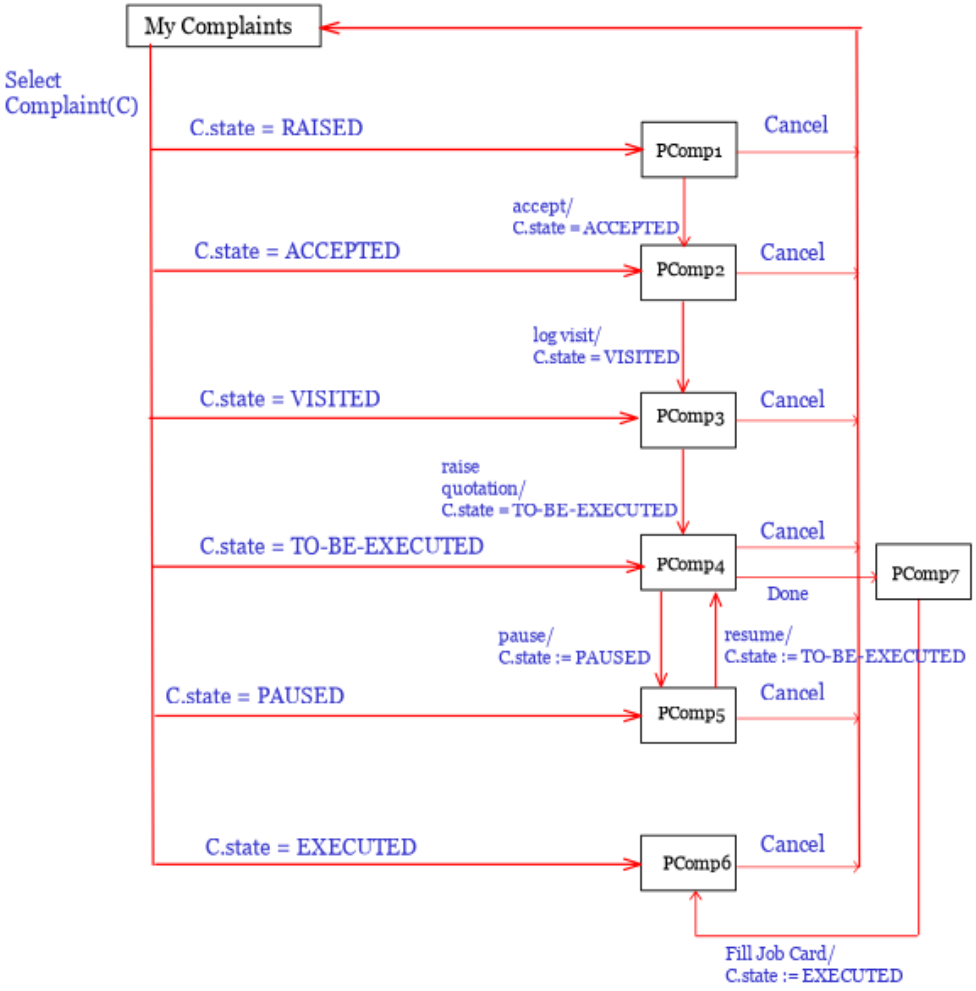
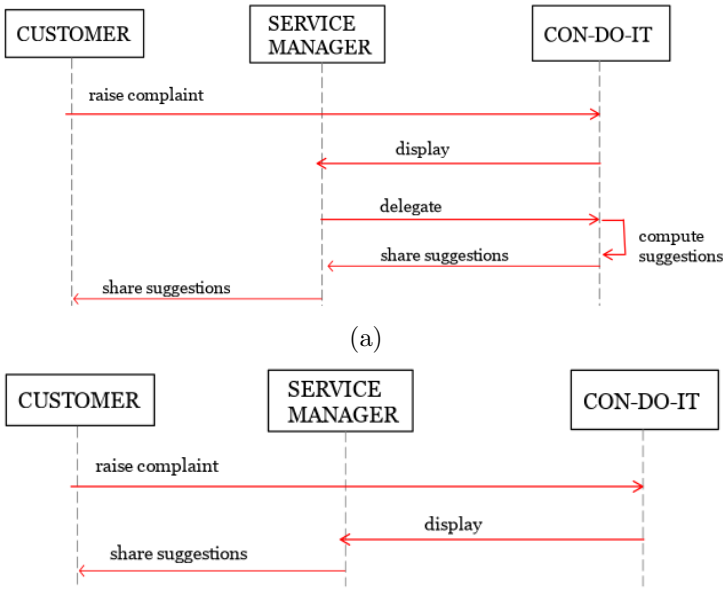


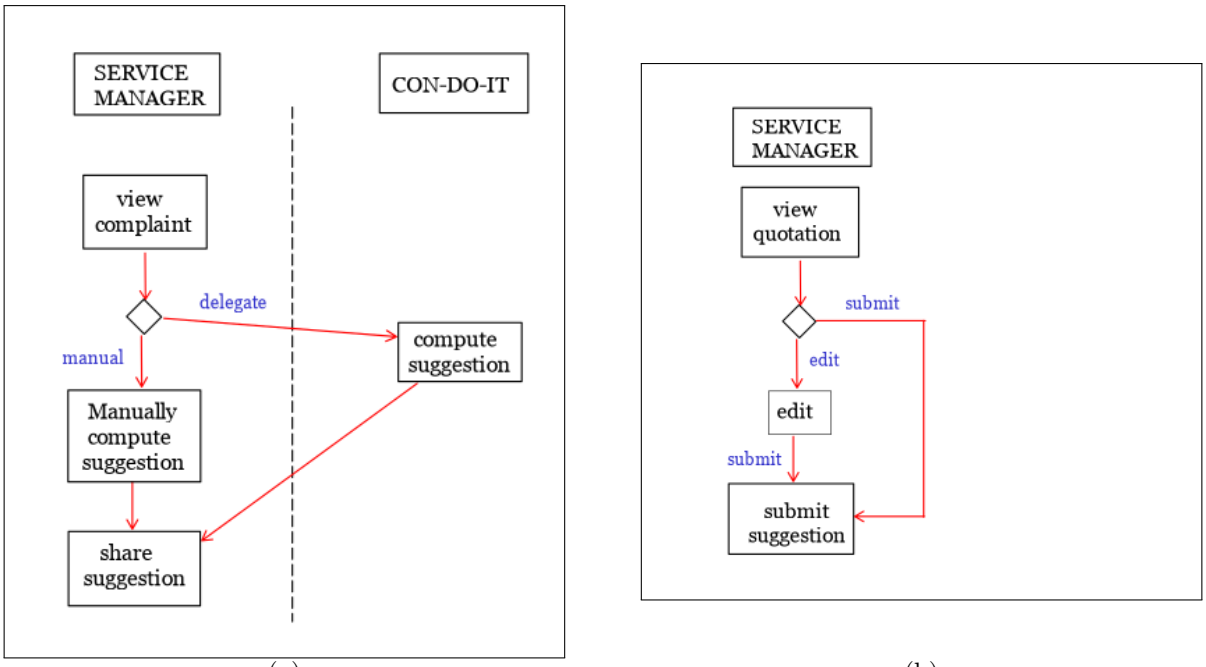
Figure 5.2: Plumber UI Navigation



(a)

(b)

Figure 5.3: Plumber Suggestion: (a) Automatic; (b) Manual

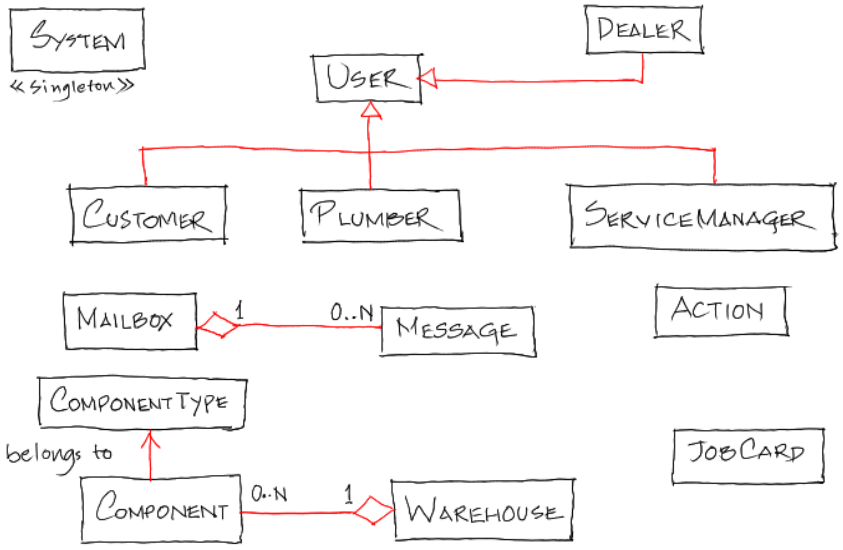


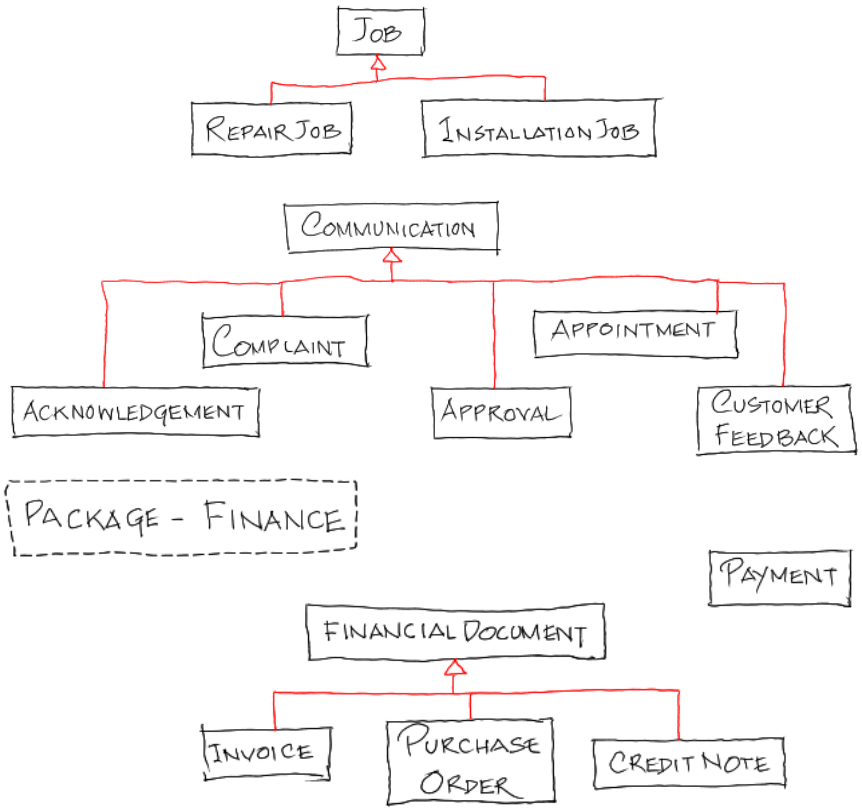
(a) (b)

Figure 5.4: Activity diagrams for plumber suggestion: (a) Automatic; (b) Manual

Chapter 6

Object Model





Future work on Object Model

Identification of additional classes, their attributes and relations. This will lead to uncovering avenues of both reuse and modularisation.

Chapter 7

Data Model

(Database design)

7.1 Collections

- Complaints
- RegisteredCustomer
- RegisteredManager
- RegisteredPlumber
- RegisteredUsers

7.2 Table Details

- **RegisteredUsers** = (username *, contact, email, firstName, lastName, password)
- **RegisteredPlumber** = (username *, acceptedComplaints array, complaints array, password, specialisations array)
- **RegisteredManager** = (username *, password)
- **RegisteredCustomer** = (username (primary key), complaints (array), contact, password)
- **Complaints** = (complaintID *, billAmount, complaint, complaintCategory array, contact, paid, plumberUsername, purchaseCostHistory array, purchaseDetailsHistory array, purchaseTypeHistory array, status, username)

Future work on data modelling

Tables need to normalised for functional independence and optimal use of space.

Chapter 8

Algorithms

8.1 Plumber Assignment

```
function Algo(){
    // JS program to implement
    // the above approach
    let maxsize = Number.MAX_VALUE

    // Stores the found edges
    let found = []

    // Stores the number of nodes
    let N = 0

    // Stores the capacity
    // of each edge
    let cap = []

    let flow = []

    // Stores the cost per
    // unit flow of each edge
    let cost = []

    // Stores the distance from each node
    // and picked edges for each node
    let dad = []
    let dist = []
    let pi = []
    let assignment = [];

    let INF = Math.floor(maxsize / 2) - 1

    // Function to check if it is possible to
    // have a flow from the src to sink
    function search(src, sink)
    {

        // Initialise found[] to false
```

```
let found = new Array(N).fill(false)

// Initialise the dist[] to INF
let dist = new Array(N + 1).fill(INF)

// Distance from the source node
dist[src] = 0

// Iterate until src reaches N
while (src != N)
{
    let best = N
    found[src] = true

    for (var k = 0; k < N; k++)
    {

        // If already found
        if (found[k])
            continue

        // Evaluate while flow
        // is still in supply
        if (flow[k][src] != 0)
        {
            // Obtain the total value
        let val = (dist[src] + pi[src] -
                    pi[k] - cost[k][src])

            // If dist[k] is > minimum value
            if (dist[k] > val)
            {
                // Update
                dist[k] = val
                dad[k] = src
            }
        }

        if (flow[src][k] < cap[src][k])
        {
            let val = (dist[src] + pi[src] -
                    pi[k] + cost[src][k])

            // If dist[k] is > minimum value
            if (dist[k] > val)
            {
                // Update
                dist[k] = val
                dad[k] = src
            }
        }

        if (dist[k] < dist[best])
            best = k
```

```
        }

        // Update src to best for
        // next iteration
        src = best
    }

    for (var k = 0; k < N; k++)
        pi[k] = Math.min(pi[k] + dist[k], INF)

    // Return the value obtained at sink
    return found[sink]
}

// Function to obtain the maximum Flow
function getMaxFlow(capi, costi,  src, sink)
{

    cap = capi
    cost = costi

    N = (capi).length
    found =  new Array(N).fill(false);

    flow = new Array(N);
    for (var i = 0; i < N; i++)
        flow[i] = new Array(N).fill(0)

    dist = new Array(N + 1).fill(INF)

    dad = new Array(N).fill(0)
    pi = new Array(N).fill(0)

    totflow = 0
    totcost = 0



    // If a path exist from src to sink
    while (search(src, sink))
    {
        let paths = [sink];
        // Set the default amount
        amt = INF
        x = sink

        while (x != src)
        {
            amt = Math.min(
                amt,
                (flow[x][dad[x]] != 0)?flow[x][dad[x]]:
                cap[dad[x]][x] - flow[dad[x]][x])
            x = dad[x]
        }
```

```
            x = sink

            while (x != src)
            {
                // if dad[x] == 0, then add the existing array into a bigger array
                paths.push(dad[x]);

                if (flow[x][dad[x]] != 0)
                {
                    flow[x][dad[x]] -= amt
                    totcost -= amt * cost[x][dad[x]]
                }
                else
                {
                    flow[dad[x]][x] += amt
                    totcost += amt * cost[dad[x]][x]
                }
                x = dad[x]
            }

            totflow += amt

            assignment.push(paths);
        }
        // Return pair total cost and sink
        return [totflow, totcost]
    }

    // Driver Code
    let s = 0
    let t = 3

    cap = [ [ 0, 4, 0, 0, 0, 3 ],
            [ 0, 0, 3, 0, 1, 0 ],
            [ 0, 0, 0, 1, 0, 0 ],
            [ 0, 0, 0, 0, 0, 0 ],
            [ 0, 0, 0, 5, 0, 0 ],
            [ 0, 0, 1, 0, 3, 0 ] ]


    cost = [[ 0, 2, 0, 0, 0, 2 ],
            [ 0, 0, 1, 0, 2, 0 ],
            [ 0, 0, 0, 2, 0, 0 ],
            [ 0, 0, 0, 0, 0, 0 ],
            [ 0, 0, 0, 1, 0, 0 ],
            [ 0, 0, 0, 0, 2, 0 ] ]


    let ret = getMaxFlow(cap, cost, s, t)


    return (assignment);
}
```

8.2 Plumber Rating

Future work on Algorithms

Neat presentation of algorithms with pseudocode, illustrative examples etc.

Part IV

Prototype System and Demo

Chapter 9

Repository and Installation

9.1 Repository

All output (documentation and code) are currently hosted in this Git repository.

9.2 Installation Instructions

Chapter 10

Using/Testing the Prototype

The demonstration is arranged as test cases. Each test case is *manually executed* starting from an initial state. The application is brought to the initial state in the *setup* step of a test case. The setup step does the following:

1. **setup DB:** Put the database in an appropriate state.
2. **Launch UI:** Launch the appropriate UI screen as the starting point of the test case.

The test launch screen (available only for demo) is as shown in Fig. 10.1.

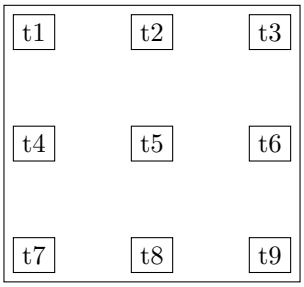


Figure 10.1: Test launch screen

10.1 Testing Using the Complaint State Machine

Here, we utilise the complaint state machine (Fig. 3.8) to design test cases. In doing so, the application must first be brought to a state where there is a customer who is about to raise a complaint. Thereafter, each test case below traverses one path leading from from the *init* state to one of the final states (*Cancelled*, *Done*, *Abandoned*).

10.1.1 Testing the complaint state machine: tsm_1

On launching tsm_1, the following setup steps take place:

1. **setup DB:** The DB contains the following:
   (a) One customer: C_1
   (b) A few plumbers: P_1, P_2, P_3
   (c) A service manager: M_1
   (d) A set of components: Co_1, Co_2, Co_3
   (e) ...
2. **Launch UI:** Dashboard of customer C_1

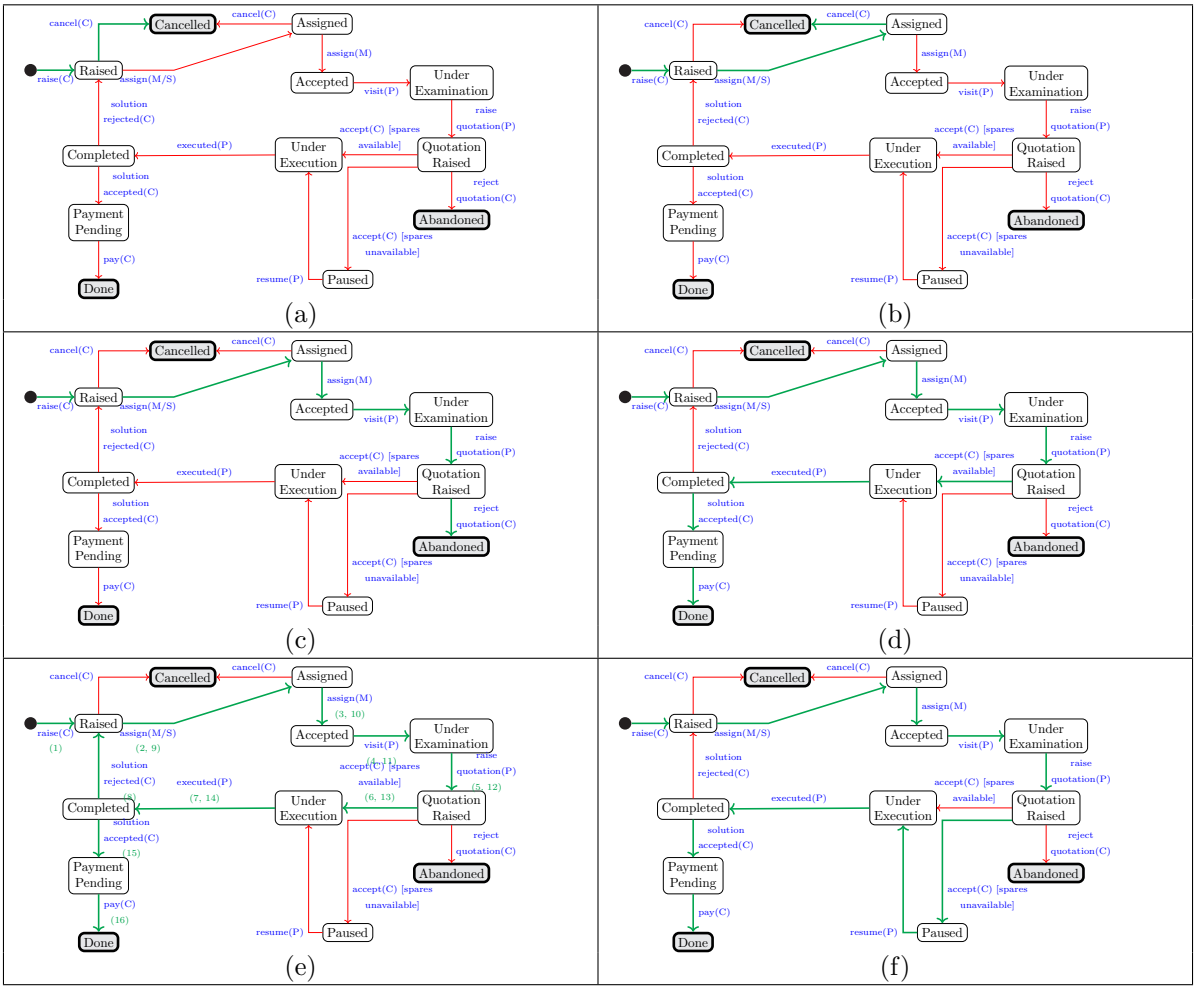


Figure 10.2: Testing the complaint state machine: (a) tsm_1; (b) tsm_2; (c) tsm_3; (d) tsm_4; (e) tsm_5; (f) tsm_6