CS4443 - Software Engineering
Professor - Dr. Manish Singh
Department of Computer Science & Engineering
Indian Institute of Technology Hyderabad

A-07 Publication Management System

Design Document

Group - 06

Kodavanti Rama Sravanth - CS20BTECH11027

Megh Shah - CS20BTECH11032

Prabhath Chellingi - CS20BTECH11038

Srivatsan T - CS20BTECH11062

April 14, 2024

Contents

1 Overview

The **Publication Management System** is a web-based software application aimed at authors, editors, reviewers, and publishers of various backgrounds and to facilitate an exchange of information about cutting-edge research across academic and industrial domains.
The Project is a Web Archive-based Publication Management System with extended support for creating and maintaining submissions of manuscripts for conferences.
In this Document, we detail the design aspects of the Code-base with an emphasis on the Module Schema and hence, the Cohesion within those Modules and the Coupling between the sets of Modules.
It is widely recognised that high Cohesion and low Coupling is very desirable in Software Design, for ease of maintenance and extensibility of the software.
The last section of the document covers the Design Specification Interface, and gives an overview of how our code-base will eventually look in the final stages of the software developmental cycle.

2 Dataflow Diagram

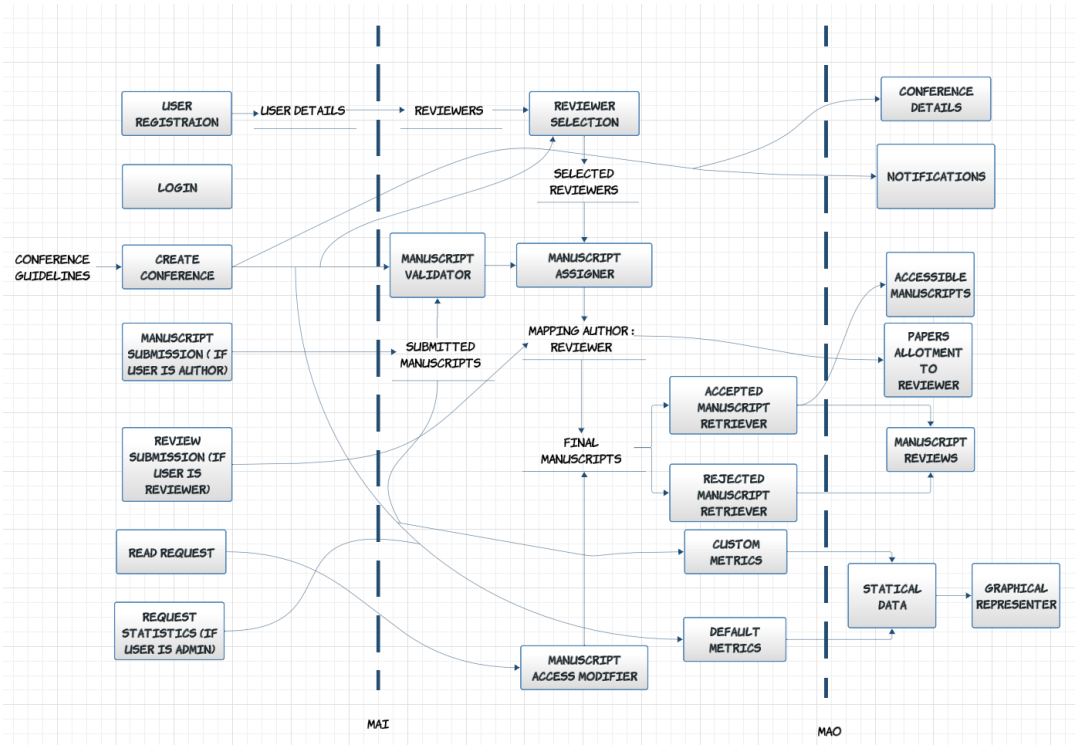

Figure 1: DFD with mai and mao divisors

3 Structured Charts

3.1 First-Level Factored Modules

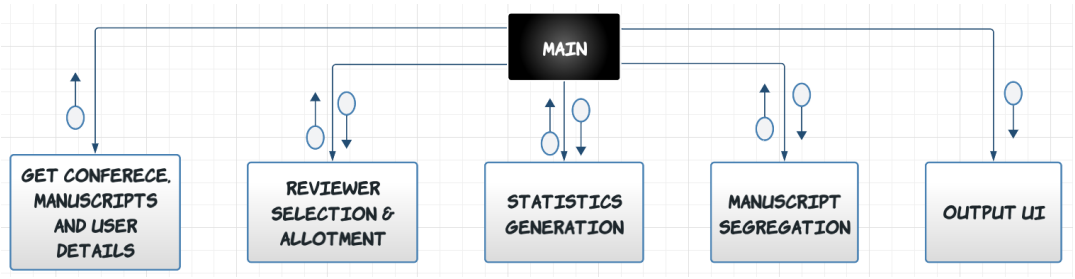

Figure 2: First-level Factored Modules

3.2 Factored Input Modules

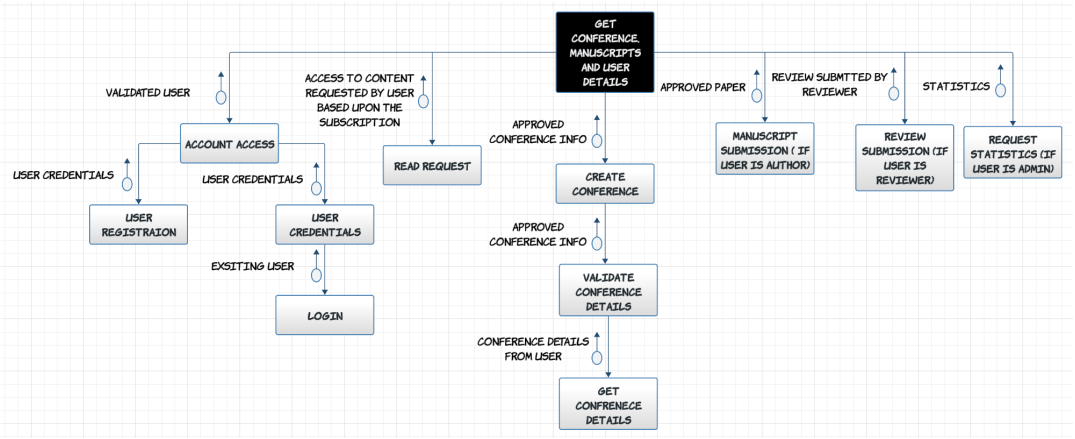

Figure 3: Factored Input Modules

3.3 Factored Transform Modules

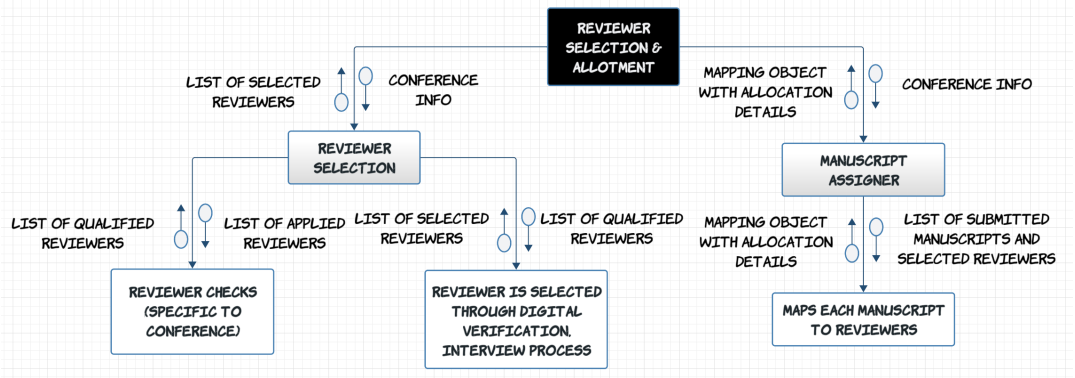

Figure 4: Central Transform (Reviewer Selection and Allotment)

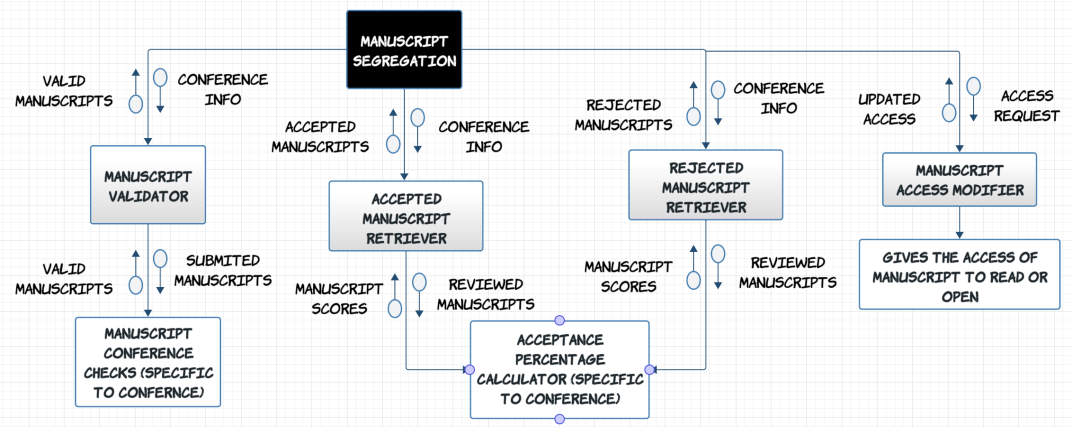

Figure 5: Central Transform (Statistics Generation)

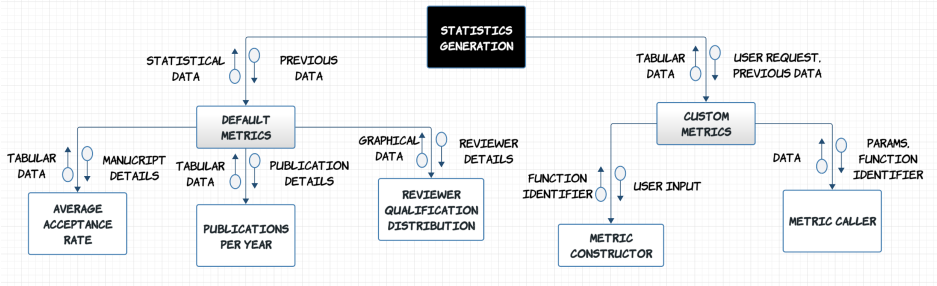

Figure 6: Central Transform (Manuscript Segregation)

3.4 Factored Output Modules

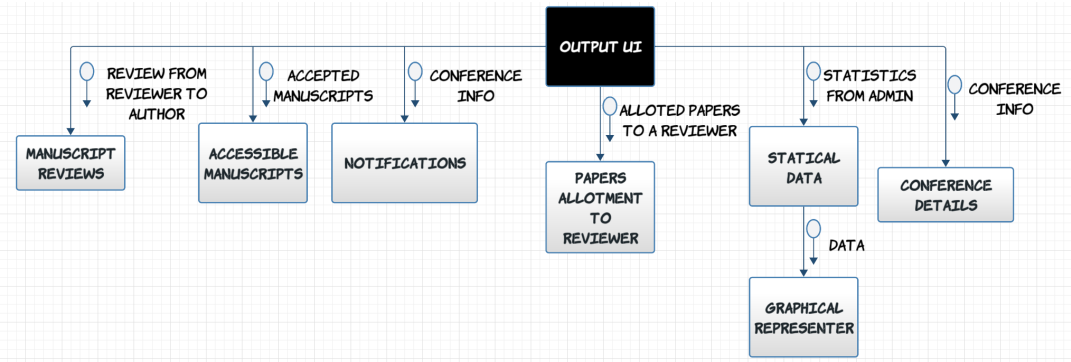

Figure 7: Factored Output Modules

3.5 Final Structured Chart

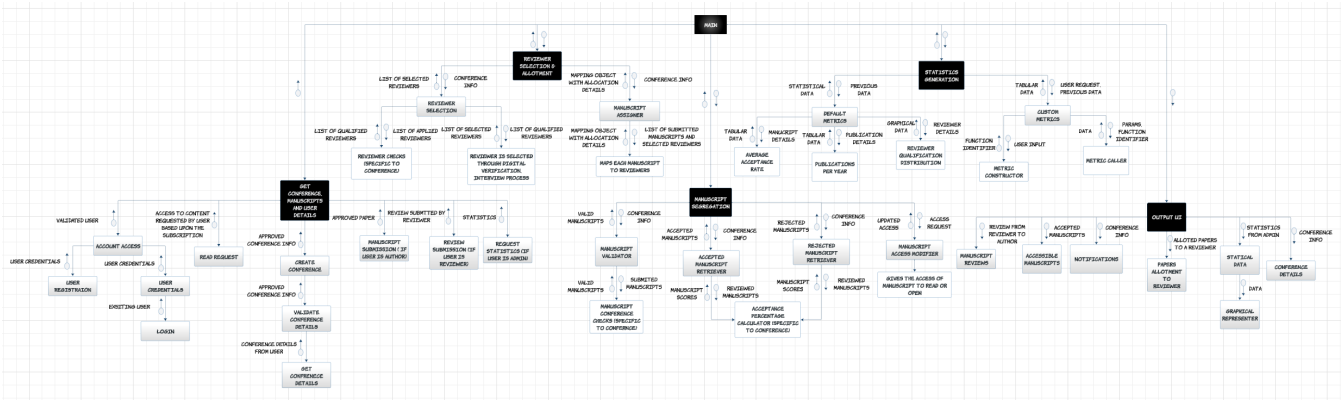

Figure 8: Final structured chart

4 Design Analysis

In this section, we discuss the details of the implementation of the selected schema of Modules.
We list out the Final Factored Modules along with their metadata.
We later discuss certain specific properties of the selected schema of Modules.
We also look at the Cohesion (intra-module code dependencies) and Coupling (inter-module code dependencies) for the selected Module Schema.

4.1 Final Factored Modules

The Table is ordered by decreasing LOC, for ease of data interpretation.

Module	Type	LOC	Cohesion Type
Manuscript Validator	Composite	500	Sequential
Reviewer Selection	Composite	500	Sequential
Manuscript Assigner	Composite	500	Functional
Accepted Manuscript Retriever	Composite	200	Sequential
Rejected Manuscript Retriever	Composite	200	Sequential
Manuscript Access Modifier	Transform	200	Procedural
Default Metrics	Composite	200	Communicational
Custom Metrics	Composite	200	Communicational
Account Access	Input	100	Communicational
Review Submission	Input	100	Functional
Request Statistics	Input	100	Functional
Manuscript Reviews	Output	100	Functional
Accessible Manuscripts	Output	100	Functional
Notifications	Output	100	Functional
Paper Allotment	Output	100	Functional
Statistics	Output	100	Functional
Conference Details	Output	100	Functional
Create Conference	Input	100	Temporal
Read Request	Input	50	Functional
Manuscript Submission	Input	50	Functional

Table 1: Final Factored Modules, ordered by LOC

4.2 Cohesion

For ease of representation, we have assigned the Degree of Coupling scores on a scale of 1 to 5, with:

- 1 - Low Coupling
- 2 - Low to Medium Coupling
- 3 - Medium Coupling
- 4 - Medium to High Coupling
- 5 - High Coupling

The Table is ordered by increasing Degree of Coupling Scores, for ease of data interpretation.

Module	Coupling Degree	Justification for Cohesion
Manuscript Access Modifier	1	It changes the access state of manuscript
Manuscript Assigner	1	It maps manuscripts to reviewers randomly
Account Access	1	It has two functions: registration and login
Read Request	1	It initiates the request for reading access of manuscripts
Manuscript Submission	1	It submits the manuscript
Review Submission	1	It submits the review
Request Statistics	1	It raises a request of statistical data
Manuscript Reviews	1	It shows the reviews
Notifications	1	It gives a notification
Manuscript Validator	2	It performs manuscript checks (conference specific) and other plagiarism checks (future version)
Create Conference	2	It creates the conference with guidelines
Custom Metrics	2	It has two functionalities to create and call metrics
Accessible Manuscripts	2	It shows the accessible manuscripts
Paper Allotment	2	It shows the reviewer and manuscript allocation
Statistics	2	produces the statistical data in the form of graphs and tables
Conference Details	2	shows the loaded conference details and guidelines
Accepted Manuscript Retriever	3	Calculation of manuscript score and acceptance of the documents
Rejected Manuscript Retriever	3	Calculation of manuscript score and rejection of the documents
Reviewer Selection	3	It validates Conference Guidelines and selects reviewers through the interview process
Default Metrics	3	It calculates multiple functions from User and System data

Table 2: Cohesion Table, ordered by Degree of Coupling

4.3 Module Count

Modules	Count
Composite	7
Input	6
Output	6
Transform	1
Coordinate	0
Total	20

Table 3: Module Count, ordered by frequency

4.4 Modules of Significant Complexity

From Table 1, we note that the following modules have the highest complexity in terms of LOCs

- Manuscript Validator (with a Coupling Degree of 2)
- Manuscript Assigner (with a Coupling Degree of 1)
- Reviewer Selection (with a Coupling Degree of 1)

It is important to have low Coupling Degrees for Modules having many lines of code, for software reliability and extensions, as stated in the Document Overview.

4.5 Modules with significant Fan-In and Fan-Out

The following is a list of high Fan-Out Modules:

- Default Metrics
- Custom Metrics
- Conference Details
- Manuscript Validator

The following is a list of high Fan-In Modules:

- Statistics
- Accepted Manuscript Retriever
- Rejected Manuscript Retriever
- Manuscript Validator

Note: A vast majority of the above Modules are ones with also high Coupling, which is to be expected as the Degree of Coupling involves both the Fan-Out and Fan-In of a Module.
Therefore, for a Module with high Degree of Coupling, we can expect it to be on at least one of the high Fan-Out or Fan-In lists.

4.6 Expected size of Software

As per out Factored Modules in Table 1, we get a direct lower bound on the number of lines of code to be 3600 LOC.
In practice, an additional 20% safety margin gives us 4320 lines of code.

5 Design Specification Interface

```
class ContactUser:
    """
        Class to represent the contact of the users in the system
    """

    phone_no : str
    email : str

class User:
    """
        Class to represent any user who has an account in the system. This will act as the parent class for the specialized users
    """

    user_id : int
    first_name : str
    last_name : str
    contact : ContactUser
    password : str

    def change_firstname(self, new_firstname : str) -> None:
        self.first_name = new_firstname

    def change_lastname(self, new_lastname : str) -> None:
        self.last_name = new_lastname

    def change_password(self, new_password : str) -> None:
        self.password = new_password

    '''
        We'll now have functions for user registration and user login
        User registration will simply be the constructor for this class or any of the
        children classes of this class
    '''
    def user_login(self, user_id : int, password : str) -> bool:
        '''
            This function verifies the password for a given user_id
        '''
        pass


class Review:
    """
        Class to store a single review for a manuscript
    """
    comments : str
    approved : bool
    points : int
```

```
48
49  class Author(User):
50      """
51          Class to represent Author
52      """
53      qualification : list[str]
54
55      def read_reviews(paper_id : int) -> list[Review]:
56          '''
57              This function would check if the paper id belongs to the authors
    and if
58              successful, lets the author read the reviews for the specific paper
    . This function is
59              a part of the Output UI
60          '''
61
62  class Reviewer(User):
63      """
64          Class to represent Reviewers
65      """
66      expertise : list[str]
67
68  class Paper:
69      """
70          Class to represent a research paper
71      """
72      paper_id : int
73      title : str
74      authors : list[Author]
75      abstract : str
76      tags : list[str]
77      manuscript : str
78
79      reviewer : list[Reviewer]
80      reviews : list[Review]
81
82      is_reviewed : bool
83      is_accepted : bool
84      is_open_to_read : bool
85
86  class Reader(User):
87      """
88          Class to represent Readers
89      """
90      is_subscribed : bool
91
92      def read_paper(self, paper_id : int) -> Paper :
93          '''
94              Function to return a manuscript requested by the user provided his
    subscription
95              status and the accessiblity status of the manuscript
96          '''
97
98      def request_paper(self, paper_id : int) -> None:
99          '''
100             Function to request a given paper to be opened if it is not alredy.
    If triggered it will be notified
101             to the admin and can be automated to be opened on multiple requests
102         '''
```

```
103
104 class Admin(User):
105     """
106         Class to represent Admin of a conference
107     """
108     pass
109
110
111 class ConferenceRule:
112     """
113         Class to define a rule for a conference
114     """
115     rule : function #Which returns a boolean value
116
117 class Conference:
118     """
119         Class to represent a conference
120     """
121     conference_id : int
122     conference_admin : Admin
123     confperence_name : str
124     is_open : bool
125     conference_deadline : Date
126
127     manuscript_rules : list[ConferenceRule]
128     reviewer_rules : list[ConferenceRule]
129
130     applied_reviewers : list[Reviewer]
131     accepted_reviewers : list[Reviewer]
132
133     applied_papers : list[Paper]
134     accepted_papers : list[Paper]
135
136     acceptance_percentage : int
137
138     '''
139         We'll now have functions to create a conference and populate the values
    from them
140         The basic blocks of the conference is to be provided by the conference
   admin which
141         includes the deadline and rules list for the same. They can also toggle
    the open nature
142         of the conference through methods
143
144         The other field like applied papers and reviewers is to be added by the
    authors
145     '''
146     def submit_paper(self, paper : Paper) -> None:
147         '''
148             Function to add a manuscript to the given conference. This is un-
   tested and
149             yet to be reviewed and stays as an element in applied_papers list
150         '''
151         pass
152
153     def apply_reviewer(self, reviewer : Reviewer) -> None:
154         '''
155             All eligible reviewers will be given provisions to add their
   profiles or objects
```

```
156             to the applied_reviewers lift before they are validated and
    assigned to papers.
157         '''
158         pass
159
160
161     '''
162         Now we'll define reviewer specfic functions which happens in a
    conference
163     '''
164     def review_paper(self,paper : Paper) -> Review:
165         '''
166             Lets the reviewer review a paper and return a review object which
    will be stored in the
167             paper and gets notified to the author and the admin of the
    conference. There will also
168             be checks to make sure that the reviewer is only reviewing the
    paper that he is assigned
169         '''
170         pass
171
172
173
174
175
176 class ReviewerSelection:
177     """
178         Class to represent reviwer selection process by the admin of the
    conference
179     """
180     conference : Conference
181
182     def perform_reviewer_checks(self) -> None:
183         '''
184             This function goes through the applied reviewers for the conference
     and populates
185             the accepted_reviewers attribute of the conference class based on
    the reviewer checks
186             imposed by the conference rules.
187             The reviewers then go through the interview process and get
    accepted
188         '''
189         pass
190
191     def assign_manuscripts(self) -> None:
192         '''
193             This function goes through each manuscripts and assigns reviewers
    for each of those papers
194             from the list of accepted_reviewers
195         '''
196         pass
197
198
199
200 class ManuscriptSegregation:
201     """
202         Class to deal with manuscript acceptance and checks which are specific
    to a conference
203     """
```

```
    conference : Conference

    def perform_manuscript_checks(self) -> None:
        '''
            This function goes through the applied papers for the conference and populates the
            segregates them based on the manuscript checks imposed by the conference.
            It then goes through the manual review process by the reviewers and then accepted_papers
            are once again segregated
        '''
        pass

    def calculate_acceptance_percentage(self) -> None:
        '''
            This function calculates the acceptance percentage for every conference. This can either be
            the admin's predefined number for the conference or can be calculated based on the number of
            submitted papers and applied reviewers
        '''
        pass

    def modify_paper_access(self) -> None:
        '''
            This function changes the open access parameter of the manuscript for a given conference
        '''
        pass

class StatisticsGeneration:
    """
        Class to generate statistics for a given conference given the data
    """
    conference : Conference

    def get_average_acceptance_rate(self) -> int:
        '''
            Goes through the entire applied and accepted manuscript lists and calculates the average acceptance rate
            for the conference
        '''
        pass

    def get_publications_per_year(self) -> int:
        '''
            Goes through the accepted papers in a given year and return the count
        '''
        pass

    def get_reviewer_qualification_distribution(self) -> graph:
        '''
            Goes through the accpeted reviewers for a given conference and returns a graph
            with the skill distribution among them
        '''
        pass
```

```
254
255
256 class OutputUI:
257     """
258         Class to maintain the Output UI for the PMS
259         Mainly consists of the admin functions and the automated notifications
260     """
261
262     conference : Conference
263
264     def allot_papers_to_reviewers(self, user_id : int, paper_id : int) -> None:
265         '''
266             Allots a user as one of the reviewers to the given paper
267         '''
268         pass
```