Chapter 4

Design of Multi-agent System for Resource Allocation and Monitoring
Chapter 4. Design of Multi-agent System for Resource Allocation and Monitoring

In Multi-agent System for Resource Allocation and Monitoring (MASRAM), three agents communicate with each other using Ontology. Ontology is used to share knowledge between sender and receiver agents when they interact with each other. Messages are passed from sender agent to receiver agent using database when agents are not alive. Sender agent keeps data in database and receiver agent on being alive picks up data from database. The messages may include request to receiver agent to perform some task or inform the receiver agent about the outcome of task. This chapter details the design of ontology, database, distributed algorithm and other processes involved.

4.1. Ontology Design

Ontology helps the designers of agent-based systems to make information understandable between agents and same is applied in MASRAM. Based on agents designed for MASRAM, four ontology sets are defined as given in table 4.1.

<table>
<thead>
<tr>
<th>Set</th>
<th>Ontology Name</th>
<th>Sender Agent</th>
<th>Receiver Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Onto-FCA-FSA</td>
<td>Facilitator Agent</td>
<td>Fund Seeker Agent</td>
</tr>
<tr>
<td>II</td>
<td>Onto-FCA-FAMA</td>
<td>Facilitator Agent</td>
<td>Fund Allocator and Monitor Agent</td>
</tr>
<tr>
<td>III</td>
<td>Onto-FSA-FCA</td>
<td>Fund Seeker Agent</td>
<td>Facilitator Agent</td>
</tr>
<tr>
<td>IV</td>
<td>Onto-FAMA-FCA</td>
<td>Fund Allocator and Monitor Agent</td>
<td>Facilitator Agent</td>
</tr>
</tbody>
</table>

Table 4.1: Ontology Sets

4.1.1. Onto-FCA-FSA

This is used when Facilitator Agent (FCA) sends message to Fund Seeker Agent (FSA), FSA interprets the contents and performs corresponding action. FCA sends six types of messages as shown in table 4.2.


Table 4.2: Ontology FCA-FSA

i. Source_type

This ontology helps agents to communicate with each other for sharing information on funding agencies who can allocate funds. Figure 4.1 shows the ontology for finding sources.

```
{Inform
  : sender (agent-identifier :name FCA@manish.1099/JADE)
  : receiver (agent-identifier :name FSA@manish.1099/JADE)
  : ontology ra-ontology
  : language fipa-sl
  : content
    (source_type
      (Nature_of_project_type: String
      Funds_required: Number
      Type_of_fund_seeker: String)
    )
}
```

Figure 4.1: Ontology- Source Types

ii. Project_proposal

FCA uses this ontology to pass the proposal submitted by Fund Seeker User (FSU) to FSA. FSA interprets the message using same ontology. In this ontology, both simple and complex attributes are used. Infrastructure available and manpower available are examples of complex attributes where sets are passed. Attributes like project identification number and seeker identification number are example of simple attributes. Figure 4.2 shows such ontology.
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECT_ID</td>
<td>Number</td>
</tr>
<tr>
<td>SEEKER_ID</td>
<td>Number</td>
</tr>
<tr>
<td>SUBMISSION_DATE</td>
<td>String</td>
</tr>
<tr>
<td>STATUS</td>
<td>String</td>
</tr>
<tr>
<td>CONTACT_PERSON</td>
<td>String</td>
</tr>
<tr>
<td>CONTACT_DESIGN</td>
<td>String</td>
</tr>
<tr>
<td>CORE_AREA</td>
<td>String</td>
</tr>
<tr>
<td>SOCIAL_RELEVANCE</td>
<td>String</td>
</tr>
<tr>
<td>TEACHING_EXP</td>
<td>Number</td>
</tr>
<tr>
<td>PROJECTS_EXP</td>
<td>Number</td>
</tr>
<tr>
<td>CONTACT_PHONE</td>
<td>Number</td>
</tr>
<tr>
<td>ALLOCATOR_ID</td>
<td>Number</td>
</tr>
<tr>
<td>PROJECT_OBJECTIVE</td>
<td>String</td>
</tr>
<tr>
<td>OBJECTIVE_TYPE</td>
<td>Number</td>
</tr>
<tr>
<td>PROJECT_SUMMARY</td>
<td>String</td>
</tr>
<tr>
<td>PROJECT_START_DATE</td>
<td>String</td>
</tr>
<tr>
<td>PROJECT_END_DATE</td>
<td>String</td>
</tr>
<tr>
<td>AGEURE_OF_PROJECT</td>
<td>Number</td>
</tr>
<tr>
<td>PROJECT_STATUS</td>
<td>Number</td>
</tr>
<tr>
<td>RESEARCH_EXP</td>
<td>Number</td>
</tr>
<tr>
<td>INT_JOURNAL</td>
<td>Number</td>
</tr>
<tr>
<td>NAT_JOURNAL</td>
<td>Number</td>
</tr>
<tr>
<td>CON_JOURNAL</td>
<td>Number</td>
</tr>
<tr>
<td>OTHER_JOURNAL</td>
<td>Number</td>
</tr>
<tr>
<td>BOOKS_PUBLISHED</td>
<td>Number</td>
</tr>
<tr>
<td>ESTABLISH_YEAR</td>
<td>Number</td>
</tr>
<tr>
<td>TITLE</td>
<td>String</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>String</td>
</tr>
<tr>
<td>LOGIN_ID</td>
<td>Number</td>
</tr>
<tr>
<td>FUND_CATEGORY_ID</td>
<td>Number</td>
</tr>
<tr>
<td>EXPECTED_OUTCOME_NT</td>
<td>Sequence</td>
</tr>
<tr>
<td>BUDGET_REQD_NT</td>
<td>Number</td>
</tr>
<tr>
<td>INFRA_AVAIL_NT</td>
<td>Sequence</td>
</tr>
<tr>
<td>MP_AVAIL_NT</td>
<td>Sequence</td>
</tr>
<tr>
<td>PROJECT_DONE_NT</td>
<td>Sequence</td>
</tr>
<tr>
<td>MILESTONE_NT</td>
<td>Sequence</td>
</tr>
<tr>
<td>METHOD_NT</td>
<td>Sequence</td>
</tr>
<tr>
<td>TECH_USED_NT</td>
<td>Sequence</td>
</tr>
<tr>
<td>RISKS_NT</td>
<td>Sequence</td>
</tr>
<tr>
<td>COST_ANALYSIS</td>
<td>Sequence</td>
</tr>
<tr>
<td>BENEFIT_ANALYSIS</td>
<td>Sequence</td>
</tr>
</tbody>
</table>

Figure 4.2: Ontology- Project Proposal
Chapter 4. Design of Multi-agent System for Resource Allocation and Monitoring

iii. **Fund utilization**

The ontology *fund utilization* is used to share the information of utilization of funds by FSU between FCA and FSA. Figure 4.3 shows the structure of this ontology.

iv. **Proposal status**

FCA inquires the status of proposal from FSA through this ontology. Figure 4.4 defines the structure of ontology used to share information between FCA and FSA.

![Figure 4.3: Ontology-Fund Utilization](image)

![Figure 4.4: Ontology-Proposal Status](image)

v. **Seeker_user_setprofile**

FCA uses this ontology to request FSA to create profile for FSU. Figure 4.5 shows such ontology.
vi. Seeker_user_updateprofile

FCA requests FSA to update particulars of fund seeker. Figure 4.6 shows structure of this ontology.

4.1.2. Onto-FCA-FAMA

This is used when FCA sends message to Fund Allocator and Monitor Agent (FAMA), FAMA interprets the contents and performs particular action. FCA sends nine different messages as shown in table 4.3.

---

Figure 4.5: Ontology-Set Profile

Figure 4.6: Ontology-Update Profile

---
Table 4.3: Ontology-FCA-FAMA

i. **Find-category**

FCA requests FAMA to provide list of funding categories using this ontology. This ontology passes the blank structure, fills contents and replies back. Figure 4.7 shows the structure of fund categories.

```
(Request
  : sender (agent-identifier :name FCA@manish:1099/JADE)
  : receiver (agent-identifier :name FAMA@manish:1099/JADE)
  : ontology ra-ontology
  : language fipa-sl
  : content (fund_category (
    FUND_ALLOCATOR_ID :Number,
    NATURE_OF_PROJECT :Number,
    DESCRIPTION :String,
    Evaluation_criteria Sequence
      (CRITERIA_ID :Number
       CRITERIA_DESC :String
       CRITERIA_TYPE :String
       MIN_REQUIRED :Number
      ),
    FUND_AVAILABLE :Number,
    PERCENTILE 1 :Number,
    MAXALLOC 1 :Number,
    PERCENTILE 2 :Number,
    MAXALLOC 2 :Number,
    PERCENTILE 3 :Number,
    MAXALLOC 3 :Number,
    MAX_FUND :Number,
    PI_WT :Number,
    NO_OF_EXPERTS :Number,
    NO_OF_FEEDBACK :Number
  )
)
```

Figure 4.7: Ontology-Find Category

56
ii. Reviewed_proposal

Reviewer user reviews the allocation made by FAMA. After review, reviewer sends the remarks on allocation to FAMA through FCA. FCA uses this ontology to pass information. Figure 4.8 shows the structure of this ontology.

```
(INFORM
  :sender (agent-identifier :name FCA@.manish:1099/JADE)
  :receiver (agent-identifier :name FAMA@.manish:1099/JADE)
  :ontology ra-ontology
  :language fipa-sl
  :content
    (Reviewed_proposal
      (PROJECT_ID: Number,
       ACT_FUNDALLOCATED: String)
    )
)
```

Figure 4.8: Ontology-Reviewed Proposal

iii. Allocator_user_setprofile

FCA requests FAMA to create login for fund allocator user. FCA passes the information of fund allocator user to FAMA using allocator_user_setprofile ontology. Figure 4.9 shows the structure of this ontology.

```
(REQUEST
  :sender (agent-identifier :name FCA@manish:1099/JADE)
  :receiver (agent-identifier :name FAMA@manish:1099/JADE)
  :ontology ra-ontology
  :language fipa-sl
  :content
    (allocator_user_setprofile
      (ALLOCATOR_ID :Number,
       ALLOCATOR_NAME :String,
       CONTACT_PERSON :String,
       ALLOCATOR_ADDRESS :String,
       ALLOCATOR_PHONE :Number,
       ALLOCATOR_EMAIL :String,
       ALLOCATOR_SPECIAL :String,
       ALLOCATOR_EXP :Number,
       ALLOCATOR_SUB_DATE :String,
       ALLOCATOR_PWD :String,
       FORM_NAME :String,
       MAX_ARBI_NO :Number)
    )
)
```

Figure 4.9: Ontology- Allocator Profile
iv. *Allocator_user_updateprofile*

FCA requests FAMA to update the particulars of fund allocator user. FCA passes the updated information of fund allocator user to FAMA using *allocator_user_updateprofile* ontology. Figure 4.10 shows the structure of this ontology.

![Figure 4.10: Ontology- Allocator Profile (Update)](image)

v. *Reviewer_user_setprofile*

FCA requests FAMA to create login for reviewer. FCA passes the information of reviewer to FAMA using *reviewer_user_setprofile* ontology. Figure 4.11 shows the structure of this ontology.

![Figure 4.11: Ontology- Reviewer Profile (Set)](image)
vi. Reviewer

FCA requests FAMA to update the particulars of the reviewer. FCA passes updated information of reviewer to FAMA using `reviewer_user_updateprofile` ontology. Figure 4.12 shows the structure of this ontology.

```
Inform
: sender (agent-identifier :name FCA@manish:1099/JADE)
: receiver (agent-identifier :name FSA@manish:1099/JADE)
: ontology ra-ontology
: language fipa-sl
: content
  (reviewer_user_updateprofile
    (REVIEWER_ID :Number
      REVIEWER_NAME :String,
      REVIEWER_EMAIL :String,
      REVIEWER_PHONES :String,
      REVIEWER_PWD :String,
      REVIEWER_EXP :Number,
      REVIEWER_SUB_DATE :String,
      REVIEWER_ADDRESS :String,
      REVIEWER_SPECIAL :String)
  )
```

Figure 4.12: Ontology- Reviewer Profile (Update)

vii. Expert

FCA requests FAMA to create login for experts. FCA passes the information of expert to FAMA using `expert_user_setprofile` ontology. Figure 4.13 shows the structure of this ontology.

```
Inform
: sender (agent-identifier :name FCA@manish:1099/JADE)
: receiver (agent-identifier :name FSA@manish:1099/JADE)
: ontology ra-ontology
: language fipa-sl
: content
  (expert_user_setprofile
    (EXPERT_ID :Number
      EXPERT_NAME :String,
      EXPERT_EMAIL :String,
      EXPERT_PHONES :String,
      EXPERT_PWD :String,
      EXPERT_EXP :Number,
      EXPERT_SUB_DATE :String,
      EXPERT_ADDRESS :String,
      EXPERT_SPECIAL :String)
  )
```

Figure 4.13: Ontology- Expert Profile (Set)
viii. Expert_user_updateprofile

FCA requests FAMA to update profile of expert users. FCA passes the information of expert to FAMA using expert_user_updateprofile ontology. Figure 4.14 shows the structure of this ontology.

```
(Inform
  :sender (agent-identifier :name FCA@manish:1099/JADE)
  :receiver (agent-identifier :name FSA@manish:1099/JADE)
  :ontology ra-ontology
  :language fipa-sl
  :content (expert_user_updateprofile
    EXPERT_ID :Number,
    EXPERT_NAME :String,
    EXPERT_EMAIL :String,
    EXPERT_PHONES :String,
    EXPERT_PWD :String,
    EXPERT_EXP :Number,
    EXPERT_SUB_DATE :String,
    EXPERT_ADDRESS :String,
    EXPERT_SPECIAL :String
  )
)
```

Figure 4.14: Ontology- Expert Profile (Set)

ix. Feedback

FCA requests FAMA to accept feedback given by experts. FCA passes the information using feedback ontology. Figure 4.15 shows the structure of this ontology.

```
(Inform
  :sender (agent-identifier :name FCA@manish:1099/JADE)
  :receiver (agent-identifier :name FSA@manish:1099/JADE)
  :ontology ra-ontology
  :language fipa-sl
  :content (feedback
    (EXPERT_ID :Number,
     (FEEDBACK_FACTOR :String,
      FEEDBACK_VALUE :String
     )
   )
  )
)
```

Figure 4.15: Ontology- Feedback

4.1.3. Onto-FSA-FCA

This is used when FSA sends message to FCA. FCA interprets the contents and performs particular action. FSA sends five messages as shown in table 4.4.
Chapter 4. Design of Multi-agent System for Resource Allocation and Monitoring

<table>
<thead>
<tr>
<th>Sender Agent: Fund Seeker</th>
<th>Receiver Agent: Facilitator</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Content Message</strong></td>
<td><strong>Act</strong></td>
</tr>
<tr>
<td>Available_source</td>
<td>Inform</td>
</tr>
<tr>
<td>Proposal_id</td>
<td>Inform</td>
</tr>
<tr>
<td>Seeker_id</td>
<td>Inform</td>
</tr>
<tr>
<td>Proposal_status</td>
<td>Inform</td>
</tr>
<tr>
<td>Project_monitoring_status</td>
<td>Inform</td>
</tr>
</tbody>
</table>

Table 4.4: Ontology – FSA-FCA

i. **Available_source (with example)**

FSA processes the request of FCA to provide the list of available sources of funding agencies and informs FCA about list using `available_source` ontology. Figure 4.16 shows such ontology as an example.

```
(Inform
  : sender (agent-identifier :name FSA@manish:1099/JADE)
  : receiver (agent-identifier :name FCA@manish:1099/JADE)
  : ontology ra-ontology
  : language fipa-sl
  : content (
    Available_source: ( NATURE_OF_PROJECT_ID : 1
      NATURE_OF_PROJECT_DESC: Information and technology
      ALLOCATOR_ID: 2001
      ALLOCATOR_DESC: Department of Information Technology
      ALLOCATOR_ADDRESS: New Delhi
      ALLOCATOR_PHONE: 0112436777
      CRITERIA: (sequence ( 1, ‘Number of Technical Staff Members’,5)
        (2, ‘Project Already Handled’, 5))
      FUND_AVAILABLE : (sequence ( 500000))
      TYPE_OFSEEKER: Individual
      MAX_FUNDS: 400000
      NO_OF_EXPERTS: 2
      NO_OF_FEEDBACK: 2 ) )
)
```

Figure 4.16: Ontology-Available Source

ii. **Proposal_id**

FSA, after saving proposal for FAMA to allocate funds, generates unique project identification number and passes the same to FCA using `proposal_id` ontology. Figure 4.17 shows the structure of this ontology.

```
(Inform
  : sender (agent-identifier :name FCA@manish:1099/JADE)
  : receiver (agent-identifier :name FSA@manish:1099/JADE)
  : ontology ra-ontology
  : language fipa-sl
  : content ( PROPOSAL_ID : Number )
)
```

Figure 4.17: Ontology-Project ID
iii. Seeker_id

FSA on the request of FCA stores particulars of FSU and generates unique fund seeker identification number and passes identification number to FCA using seeker_id. Figure 4.18 describes the structure of such ontology.

```
(Inform
 :sender (agent-identifier :name FCA@manish:1099/JADE)
 :receiver (agent-identifier :name FSA@manish:1099/JADE)
 :ontology ra-ontology
 :language fipa-sl
 :content
 ( SEEKER_ID :Number
 )
)
```

Figure 4.18: Ontology-Seeker ID

iv. Proposal_status

FSA on the request of FCA informs status of proposal regarding fund allocation. Figure 4.19 describes the structure of proposal_status ontology.

```
(Inform
 :sender (agent-identifier :name FCA@manish:1099/JADE)
 :receiver (agent-identifier :name FSA@manish:1099/JADE)
 :ontology ra-ontology
 :language fipa-sl
 :content
 ( PROJECT_ID :Number,
 STATUS :String,
 RANK_DATE :String,
 WEIGHT :Number,
 BUDGET_REQD :Number,
 BUDGET_ALLOCATED :Number,
 ACT_FUND_ALLOCATED :Number
 )
)
```

Figure 4.19: Ontology-Proposal Status

v. Project_monitoring_status

FSA on the request of FCA informs the status of project with respect to continuity of project. Figure 4.20 describes structure of project_monitoring_status ontology.
4.1.4. Onto-FAMA-FCA

This is used when FAMA sends message to FCA, FCA interprets the contents and performs the particular action. FAMA sends five messages as shown in table 4.5.

<table>
<thead>
<tr>
<th>Content Message</th>
<th>Act</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allotted_allocator_id</td>
<td>Inform</td>
</tr>
<tr>
<td>Allotted_reviewer_id</td>
<td>Inform</td>
</tr>
<tr>
<td>Allotted_fund_category</td>
<td>Inform</td>
</tr>
<tr>
<td>Review_fund_allocation</td>
<td>Inform</td>
</tr>
<tr>
<td>Expert_feedback_due</td>
<td>Inform</td>
</tr>
</tbody>
</table>

Table 4.5: Ontology FAMA-FCA

i. Fund_allocator_id

FAMA informs FCA about identification number generated for allocator using fund_allocator_id ontology. Figure 4.21 shows structure of this ontology.

ii. Allotted_reviewer_id

FAMA, after generating reviewer identification number, informs FCA using allotted_reviewer_id ontology. On the same pattern expert ID is generated and informed. Figure 4.22 shows structure of this ontology.
iii. Allotted_fund_category

FAMA, after generating fund category identification number, informs FCA using allotted_fund_category ontology. Figure 4.23 shows the structure of this ontology.

```plaintext
(Inform
 : sender (agent-identifier :name FAMA@manish:1099/JADE)
 : receiver (agent-identifier :name FCA@manish:1099/JADE)
 : ontology ra-ontology
 : language fipa-sl
 : content
 ( ALLLOTTED_FUND_CATEGORY: Number )
)
```

Figure 4.23: Ontology-Allotted Fund Category

iv. Review_fund_allocation

FAMA informs reviewer through FCA the allocation made by the system using review_proposal ontology. Figure 4.24 shows the structure of this ontology.

```plaintext
(Inform
 : sender (agent-identifier :name FAMA@manish:1099/JADE)
 : receiver (agent-identifier :name FCA@manish:1099/JADE)
 : ontology ra-ontology
 : language fipa-sl
 : content
 ( Sequence ( PROJECT_ID : Number ,
 TITLE : String,
 WEIGHT : Number,
 FUNDS_DEMANDED : Number,
 FUNDS_ALLOCATED : Number,
 ACTUAL_FUNDS_TO_ALLOCATE : Number ))
)
```

Figure 4.24: Ontology-Review Fund Allocation
v. Project_monitoring_due

FAMA informs experts through FCA the projects due for providing feedback on progress. Figure 4.25 shows the structure of this ontology.

```
(Inform
  : sender (agent-identifier :name FAMA@manish:1099/JADE)
  : receiver (agent-identifier :name FCA@manish:1099/JADE)
  : ontology ra-ontology
  : language fipa-sl
  : content ( Sequence (
    PROJECT_ID  : Number,
    TITLE       : String
  ))
)
```

Figure 4.25: Ontology-Project Monitoring Due

### 4.2. Database Design

Agents of MASRAM are mapped into schema objects of Object Relational Database Management System (ORDBMS). Actions of Agents are mapped into methods/stored procedures of schema objects. Following broad database objects are identified for database support.

i. Fund Seeker Object

Fund Seeker object is designed to store the information of the fund seeker user. It stores the complete profile of the fund seeker user including login credentials. This object also helps in validating the user.

ii. Fund Allocator Object

Fund Allocator Object stores the profile of fund allocating agencies along with login credentials. It stores information on nature of project for which funds are being granted, type of fund seeker, maximum limit, percentile policy and amount of funds to be funded.

iii. Proposals Object

Proposals Object is used to store detailed proposal, its status such as fund allocated, rejected etc. It is linked with all the other objects in database.

iv. Allocation Object

This object keeps the information of evaluation process. Information like weight assigned, proposals’ comparison, reviewers’ remarks and fund allocated is stored. Step wise outcome of allocation procedure (algorithm that is used in allocating funds) is also stored in this object.
v. Feedback Object

This object keeps information of feedback provided by experts based on presentation given by fund seeker before allocation and presentation given during project execution (mid-term and final).

vi. Utilization Object

Once the funds are allocated to the proposals, the status of utilization of funds from time to time by fund seeker is stored in this object. The aim is to know fulfillment of the objective for which funds were granted.

Based on above requirement, following database schema objects are designed.

I. Tables/Nested Tables

II. Views

III. Stared Procedures/Functions

4.2.1. Tables/Nested Tables

Table 4.6 lists the tables designed to store information/data related to fund seekers, fund allocators, experts, reviewers, proposals and status thereafter. The list of nested tables designed is given in table 4.7.

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>PROJECT_PROPOSALS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOCATION</td>
<td>PROJECT_PROPOSALS</td>
</tr>
<tr>
<td>ALLOCATOR_FREQUENCY_MAS</td>
<td>PROPOSALS_PROCESSING</td>
</tr>
<tr>
<td>ALLOCATOR_MAS</td>
<td>QUALIFICATION_MAS</td>
</tr>
<tr>
<td>BUDGET_ITEM_MAS</td>
<td>QUALIFYING_CRITERIA_MAS</td>
</tr>
<tr>
<td>DESIGNATION_MAS</td>
<td>REVIEWER_ALLOCATOR_MAS</td>
</tr>
<tr>
<td>DM_DETAIL</td>
<td>REVIEWER_MAS</td>
</tr>
<tr>
<td>DM_EVALUATION</td>
<td>RISK_INDICATOR_MAS</td>
</tr>
<tr>
<td>DM_MAIN</td>
<td>RISK_INDICATOR_VALUE_MAS</td>
</tr>
<tr>
<td>FUND_ALLOCATION</td>
<td>SEEKER_MAS</td>
</tr>
<tr>
<td>FUND_CATEGORY</td>
<td>STATUS_MAS</td>
</tr>
<tr>
<td>KEYWORDS</td>
<td>WEIGHT_DETAIL</td>
</tr>
<tr>
<td>NATURE_OF_PROJECT_MAS</td>
<td>WEIGHT_MAIN</td>
</tr>
<tr>
<td>OBJECTIVE_TYPE_MAS</td>
<td>OUTCOME_MAS</td>
</tr>
</tbody>
</table>

Table 4.6: List of Tables
4.2.2. Views

The main advantage of using views is to write complex query once and fetch the data using views many times. In database design, 21 views are also designed to ease the complex queries as listed in Table 4.8.

<table>
<thead>
<tr>
<th>LIST OF VIEWS</th>
<th>VIEW_MONITING</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIEW_BUDGET_REQD</td>
<td></td>
</tr>
<tr>
<td>VIEW_CORE_AREA_INDIVIDUAL</td>
<td>VIEW_MP_EVALUATION</td>
</tr>
<tr>
<td>VIEW_DAYS_ELAPSED</td>
<td>VIEW_MP_TOTAL</td>
</tr>
<tr>
<td>VIEW_DETAIL_WT</td>
<td>VIEW_OUTCOME</td>
</tr>
<tr>
<td>VIEW_ECO</td>
<td>VIEW_PROJECT_FATE</td>
</tr>
<tr>
<td>VIEW_EXP_INDIVIDUAL</td>
<td>VIEW_RANKING</td>
</tr>
<tr>
<td>VIEW_FUNDS_ALLOTTED</td>
<td>VIEW_REJECT</td>
</tr>
<tr>
<td>VIEW_FUND_DATES</td>
<td>VIEW_REPORT_STATUS</td>
</tr>
<tr>
<td>VIEW_INFRA_EVAL</td>
<td>VIEW_RISK_EVALUATION</td>
</tr>
<tr>
<td>VIEW_INFRA_EVALUATION</td>
<td>VIEW_WEIGHT_MAIN</td>
</tr>
<tr>
<td>VIEW_INFRA_ITEM_COUNT</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.8: List of Views

4.2.3. Stored Procedures/Functions

A stored procedure/function is precompiled set of SQL (Structured Query Language) statements. Another advantage of using stored procedure/functions is that business logic can be implemented in stored procedures and can be called in calling
program. Stored procedures/functions are clubbed together into package. Table 4.9 lists the procedures and functions.

<table>
<thead>
<tr>
<th>Procedure/Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allocate_fund_main</td>
<td>function to allocate fund_main(sls varchar2) return varchar2;</td>
</tr>
<tr>
<td>match_criteria</td>
<td>function match_criteria(all_id number, pid number) return number;</td>
</tr>
<tr>
<td>process_insert</td>
<td>procedure process_insert(p_item varchar2, p_item as proposal varchar2, p_item_reqd varchar2, p_weight number, p_remarks varchar2, p_project_id number, p_status char, p_item_type char);</td>
</tr>
<tr>
<td>update_status_criteria</td>
<td>procedure update_status_criteria(pid number);</td>
</tr>
<tr>
<td>evaluate_core_area</td>
<td>procedure evaluate_core_area(pid number, all_id number);</td>
</tr>
<tr>
<td>evaluate_hr_development</td>
<td>procedure evaluate_hr_development(pid number, all_id number);</td>
</tr>
<tr>
<td>evaluate_national_development</td>
<td>procedure evaluate_national_development(pid number, all_id number);</td>
</tr>
<tr>
<td>evaluate_tech_avail</td>
<td>procedure evaluate_tech_avail(pid number, all_id number);</td>
</tr>
<tr>
<td>evaluate_suc_prob</td>
<td>procedure evaluate_suc_prob(pid number, all_id number);</td>
</tr>
<tr>
<td>evaluate_cost_involved</td>
<td>procedure evaluate_cost_involved(pid number, all_id number);</td>
</tr>
<tr>
<td>evaluate_eco_benefit</td>
<td>procedure evaluate_eco_benefit(pid number, all_id number);</td>
</tr>
<tr>
<td>evaluate_inra_avail</td>
<td>procedure evaluate_inra_avail(pid number, all_id number);</td>
</tr>
<tr>
<td>evaluate_mangt_cap</td>
<td>procedure evaluate_mangt_cap(pid number, all_id number);</td>
</tr>
<tr>
<td>evaluate_staff_expt</td>
<td>procedure evaluate_staff_expt(pid number, all_id number);</td>
</tr>
<tr>
<td>evaluate_project_completion</td>
<td>procedure evaluate_project_completion(pid number, all_id number, p_of_suc number);</td>
</tr>
<tr>
<td>evaluate_risk_analysis</td>
<td>procedure evaluate_risk_analysis(pid number, all_id number);</td>
</tr>
<tr>
<td>allocate_funds</td>
<td>procedure allocate_funds(all_id number, cate_id number);</td>
</tr>
<tr>
<td>countwords</td>
<td>function countwords(flag number, pid number, alloc_id number, dmain_id number, ddetail_id number) return number;</td>
</tr>
<tr>
<td>calculate_weight</td>
<td>procedure calculate_weight(all_id number, cate number);</td>
</tr>
<tr>
<td>get_sum</td>
<td>function get_sum(dm number, dd number) return number;</td>
</tr>
<tr>
<td>allocate_fund_percentile</td>
<td>procedure allocate_fund_percentile(all_id number, cate number);</td>
</tr>
</tbody>
</table>

Table 4.9: List of Stored Procedures/Functions

4.3. Distributed Algorithm

An instance of the Resource Allocation Problem of funding can be characterized by a 6-tuple \( (R, A, X, EW, CMIN, F) \) entity. In detail, the problem consists of set of consumable and available fund resources \( R \).

\[
R \rightarrow \text{Set of available resources (funds)}
\]
Chapter 4. Design of Multi-agent System for Resource Allocation and Monitoring

\{R_1, R_2, R_3 \ldots R_m\}

Where \(R_j, 1 \leq j \leq m\) belongs to \(j^{th}\) category of funds from where funds are to be allocated and \(m\) be the different categories of funds.

Set \(A\) describes the set of proposals (received from funds seekers) requiring funds to implement their projects.

\(A\rightarrow\) Set of applicants

\{A_1, A_2, A_3 \ldots A_n\}

Where \(A_i, 1 \leq i \leq n\) is \(i^{th}\) proposal submitted by fund seeker to whom the funds are to be allocated with the assumption that there are \(n\) proposals submitted by the different fund seekers.

The matrix \(X\) defines the amount of funds required by \(i^{th}\) fund seeker from \(j^{th}\) fund category where \(1 \leq i \leq n\) and \(1 \leq j \leq m\).

\[
X_{ij} = \begin{bmatrix}
x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,m} \\
x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,m} \\
x_{3,1} & x_{3,2} & x_{3,3} & \cdots & x_{3,m} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
x_{n,1} & x_{n,2} & x_{n,3} & \cdots & x_{n,m}
\end{bmatrix}
\]

Set \(EW\) is the set of expert weights

\{EW_1, EW_2, EW_3 \ldots EW_{12}\}

Let \(r\) be the total number of decision-making factors i.e. 12.

Let \(F\) be a four-dimensional array consisting of \(l, i, j\) and \(k\) dimensions. Where \(1 \leq l \leq m\), \(1 \leq i \leq n\), \(1 \leq j \leq e\), \(1 \leq k \leq ev\), \(e\) is number of experts and \(ev\) is scale provided by expert.

Let \(C_{\text{MIN}}(i,k)\) be criteria set by allocator for \(i^{th}\) fund category and \(k^{th}\) decision-making factor. Table 4.10 describes weight-based algorithm designed to allocate funds at abstract level.
Chapter 4. Design of Multi-agent System for Resource Allocation and Monitoring

Gathering Initial Values
This step is concerned with gathering information from the environment. This includes proposals submitted by the fund seekers, availability of funds, experts’ feedback on presentation, criteria to qualify for funding, fund seeker profile, fund allocator profile, expert profile, reviewer profile and weights of decision-making factors. FSA provides information on proposal submission and profile of the fund seeker user. FAMA provides the remaining information.

Filtering Proposals (Matching Criteria)
This step extracts the information from the proposals, matches with the qualifying criteria set by fund allocator and filters the proposals. The proposals satisfying criteria are made eligible to avail funds; remaining proposals are rejected. FAMA checks the minimum eligibility criteria.

Reviewing Proposals
In some cases, especially in minor and short duration projects, the fund seekers are required to present the proposals before experts. The feedback of these experts is compiled. The proposals satisfying the minimum qualifying scale set by the allocator are considered for funding. Each expert is provided a checklist where he/she gives answer.

Evaluating Proposals
This step evaluates the proposals against each of the twelve decision-making factors. Out of the 12 factors, factor numbers C1.1, C1.3 and C4.2 are evaluated based on keywords provided by the fund allocator. Factor number C1.2, C2.2, C3.1, C3.2, C4.3, C511, and C5.2 are evaluated based on information provided by individual fund seeker in the proposal. Factor number C2.1 and C4.1 are evaluated by comparing proposals with one another. Each factor is given a numeric value, also called relative scale based on evaluation. After factor level evaluation, AHP technique is applied to compare proposals with one another.

Allocating Funds
a) Assigning Expert Weights
Each factor calculated in step IV is multiplied by their respective weight suggested by experts, and Geometric Mean (GM) of the weighted values of all the factors corresponding to individual proposal is calculated. This gives a single numeric value, a relative one to proposal and helps in ranking the proposals.

b) Calculating Cut offs
Based on the percentile policy set by the fund allocator and weights calculated during execution of step V(a), three different cut offs are calculated. The fund allocator also sets the maximum percentage of demanded funds that can be given within each percentile. Cut offs are then calculated. During allocation, some proposals may be given 100 percent funds as demanded; some may be given part of the demanded budget while others may be denied funds due to having weighted relative scale below the last cut off.

c) Distribution of Funds
If the sufficient funds are available, proposals falling under different cut offs are given funds accordingly subject to the maximum limit set by fund allocator. The proposals having weight less than the last cut off are denied funds.

Table 4.10: Weight-based Algorithm at Abstract Level

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Gathering Initial Values</td>
</tr>
<tr>
<td></td>
<td>This step is concerned with gathering information from the environment. This includes proposals submitted by the fund seekers, availability of funds, experts' feedback on presentation, criteria to qualify for funding, fund seeker profile, fund allocator profile, expert profile, reviewer profile and weights of decision-making factors. FSA provides information on proposal submission and profile of the fund seeker user. FAMA provides the remaining information.</td>
</tr>
<tr>
<td>II</td>
<td>Filtering Proposals (Matching Criteria)</td>
</tr>
<tr>
<td></td>
<td>This step extracts the information from the proposals, matches with the qualifying criteria set by fund allocator and filters the proposals. The proposals satisfying criteria are made eligible to avail funds; remaining proposals are rejected. FAMA checks the minimum eligibility criteria.</td>
</tr>
<tr>
<td>III</td>
<td>Reviewing Proposals</td>
</tr>
<tr>
<td></td>
<td>In some cases, especially in minor and short duration projects, the fund seekers are required to present the proposals before experts. The feedback of these experts is compiled. The proposals satisfying the minimum qualifying scale set by the allocator are considered for funding. Each expert is provided a checklist where he/she gives answer.</td>
</tr>
<tr>
<td>IV</td>
<td>Evaluating Proposals</td>
</tr>
<tr>
<td></td>
<td>This step evaluates the proposals against each of the twelve decision-making factors. Out of the 12 factors, factor numbers C1.1, C1.3 and C4.2 are evaluated based on keywords provided by the fund allocator. Factor number C1.2, C2.2, C3.1, C3.2, C4.3, C511, and C5.2 are evaluated based on information provided by individual fund seeker in the proposal. Factor number C2.1 and C4.1 are evaluated by comparing proposals with one another. Each factor is given a numeric value, also called relative scale based on evaluation. After factor level evaluation, AHP technique is applied to compare proposals with one another.</td>
</tr>
<tr>
<td>V</td>
<td>Allocating Funds</td>
</tr>
<tr>
<td>a)</td>
<td>Assigning Expert Weights</td>
</tr>
<tr>
<td></td>
<td>Each factor calculated in step IV is multiplied by their respective weight suggested by experts, and Geometric Mean (GM) of the weighted values of all the factors corresponding to individual proposal is calculated. This gives a single numeric value, a relative one to proposal and helps in ranking the proposals.</td>
</tr>
<tr>
<td>b)</td>
<td>Calculating Cut offs</td>
</tr>
<tr>
<td></td>
<td>Based on the percentile policy set by the fund allocator and weights calculated during execution of step V(a), three different cut offs are calculated. The fund allocator also sets the maximum percentage of demanded funds that can be given within each percentile. Cut offs are then calculated. During allocation, some proposals may be given 100 percent funds as demanded; some may be given part of the demanded budget while others may be denied funds due to having weighted relative scale below the last cut off.</td>
</tr>
<tr>
<td>c)</td>
<td>Distribution of Funds</td>
</tr>
<tr>
<td></td>
<td>If the sufficient funds are available, proposals falling under different cut offs are given funds accordingly subject to the maximum limit set by fund allocator. The proposals having weight less than the last cut off are denied funds.</td>
</tr>
</tbody>
</table>
Chapter 4: Design of Multi-agent System for Resource Allocation and Monitoring

The algorithm allocates the fund category wise. Table 4.11 shows the pseudo of steps mentioned above.

<table>
<thead>
<tr>
<th>Main Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>For (1 = 1) to (m) |</td>
</tr>
<tr>
<td>Begin |</td>
</tr>
<tr>
<td>Perform Step I |</td>
</tr>
<tr>
<td>Perform Step II |</td>
</tr>
<tr>
<td>Perform Step III |</td>
</tr>
<tr>
<td>Perform Step IV |</td>
</tr>
<tr>
<td>Perform Step V(a) |</td>
</tr>
<tr>
<td>Perform Step V(b) |</td>
</tr>
<tr>
<td>Perform Step V(c) |</td>
</tr>
<tr>
<td>End |</td>
</tr>
</tbody>
</table>

Table 4.11: Weight-based Algorithm-Main

Each step of the algorithm is further detailed as a pseudo code. Table 4.12 describes the Step I in detail i.e. gathering initial values.

<table>
<thead>
<tr>
<th>Gathering Initial Values (Step I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let (R = {R_1, R_2, R_3, \ldots R_m}) be set of resources (funds) for a particular category of funds. |</td>
</tr>
<tr>
<td>Let (A_n, 1 &lt;= i &lt;= n) be n fund seekers requiring funds. |</td>
</tr>
<tr>
<td>Let (X_{ij}) be the funds requirement by (i^{th}) fund seeker from (j^{th}) fund category. |</td>
</tr>
<tr>
<td>Let (r) be number of Decision Making Factors (DMF). |</td>
</tr>
<tr>
<td>Let (\text{CMIN}(i, j, k), 1 &lt;= i &lt;= r) be minimum requirement for (i^{th}) criteria. |</td>
</tr>
<tr>
<td>Let (C_{ij}) be the criteria value of (i^{th}) fund seeker (j^{th}) criteria. (1 &lt;= i &lt;= m), (1 &lt;= j &lt;= r). |</td>
</tr>
<tr>
<td>Let (\text{EVAL}(i, j, k)) be evaluation 3-dimensional array where (i &lt;= i &lt;= m), (1 &lt;= j &lt;= n) and (1 &lt;= k &lt;= r). |</td>
</tr>
<tr>
<td>Let (AR(i, j)) be fund allotted matrix to (i^{th}) fund seeker from (j^{th}) resource. |</td>
</tr>
<tr>
<td>Let (\text{PROJECT_STATUS}(i, j)) stores status of the proposals, whether proposal is qualified for allocation or not and where (i &lt;= i &lt;= m), (1 &lt;= j &lt;= n) |</td>
</tr>
<tr>
<td>Let (\text{perl} = \text{percentile 1}) |</td>
</tr>
<tr>
<td>Let (\text{pl} = \text{perl}) |</td>
</tr>
<tr>
<td>Let (\text{max1} = \text{maximum amount against percentile 1}) |</td>
</tr>
<tr>
<td>Let (\text{per2} = \text{percentile 2}) |</td>
</tr>
<tr>
<td>Let (\text{p2} = \text{per2}) |</td>
</tr>
<tr>
<td>Let (\text{max2} = \text{maximum amount against percentile 2}) |</td>
</tr>
<tr>
<td>Let (\text{per3} = \text{percentile 3}) |</td>
</tr>
<tr>
<td>Let (\text{p3} = \text{per3}) |</td>
</tr>
</tbody>
</table>

Table 4.12: Weight-based Algorithm (Step I)

Second step checks the minimum eligibility criteria set by the fund allocator. Table 4.13 describes the steps carried out to check the minimum eligibility criteria. Proposals who qualify for the minimum eligibility criteria would only be considered for fund allocation provided funds are available and proposals get relative scale within cut offs.
Filtering Proposal - Matching Criteria (Step II)

For \( i = 1 \) to \( n \)
Begin
If(proposal(l,i) = 0) continue;
Else    Flag=0
For \( k = 1 \) to \( r \)
Begin
    If (CMIN(i,k) != cv(j,k))flag = 1
End for
If (flag = 0) project_status(l,i) = 'Qualified'
Else project_status(l,i) = 'Rejected'
End if
End if
End for

Table 4.13: Weight-based Algorithm (Step II)

Reviewing proposals is done based on the experts’ feedback given after the fund seeker has presented the proposal. This exercise is not applicable to every type of fund category. It is up to the fund allocator to define. If it is not required for particular category, proposals falling under that category are supposed to have been qualified for funding. However, for those categories where presentation is must, review of proposal is done. Feedback from the experts is taken and compiled. Proposal must qualify for availing funds. Table 4.14 describes pseudo code for review of proposals.

Reviewing Proposals (Step III)

If pre-allocation flag of \( i^{th} \) category is false then
   For \( (i = 1 \) to \( n \) )
   Begin
      Proposal(l,i) = 1
   End for
Else
   For \( (i = 1 \) to \( n \) )
   Begin
      Cnt =0;
      For \( (j = 1 \) to \( p \) )
      Begin
         Sum =0
         For \( (k = 1 \) to \( ev \) )
         begin
            Sum += f(l,i,j,k)
         End for
         Avg = sum /ev
         If (avg <= ev/2
         Cnt ++;
         End if
      End for
      If (cnt > 0) proposal(l,i) = 1
      Else proposal(l,i) = 0;
   End for
End if

Table 4.14: Weight-based Algorithm (Step III)

Table 4.15 describes the logical flow to evaluate the qualified proposals against the twelve decision-making factors. The further detail is given in table 4.16.
Chapter 4. Design of Multi-agent System for Resource Allocation and Monitoring

### Evaluating Proposals (Step IV)

For \( I = 1 \) to \( n \)

Begin

For \( j = 1 \) to \( r \)

Begin

If (project_status(\( l, i \)) = 'Qualified') evaluate_cri(\( l, j, i \))

End if

End for

End for

---

Table 4.15: Weight-based Algorithm (Step IV)

<table>
<thead>
<tr>
<th>Procedure evaluate_cri(( i, j, k ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin</td>
</tr>
<tr>
<td>If (( j = 1 )) then</td>
</tr>
<tr>
<td>Find total number of core keywords in ( k )th proposal and let it be ( kw )</td>
</tr>
<tr>
<td>EVAL(( i, j, k )) = ( kw )</td>
</tr>
<tr>
<td>Else if (( j = 2 )) then</td>
</tr>
<tr>
<td>Find total number candidates to be trained in ( k )th proposal and let it be ( kw ) in case quantifiable outcome is given, otherwise assign 1 to ( kw )</td>
</tr>
<tr>
<td>EVAL(( i, j, k )) = ( kw )</td>
</tr>
<tr>
<td>Else if (( j = 3 )) then</td>
</tr>
<tr>
<td>Find total number national development and social impact keywords in ( k )th proposal and let it be ( kw )</td>
</tr>
<tr>
<td>EVAL(( i, j, k )) = ( kw )</td>
</tr>
<tr>
<td>Else if (( j = 4 )) then</td>
</tr>
<tr>
<td>Find total number technical used keywords in ( i )th proposal and let it be ( kw )</td>
</tr>
<tr>
<td>EVAL(( i, j, k )) = ( kw )</td>
</tr>
<tr>
<td>Else if (( j = 5 )) then</td>
</tr>
<tr>
<td>Let ( tpc ) be the total projects completed by ( k )th fund seeker</td>
</tr>
<tr>
<td>Let ( tpd ) be the total projects delayed by ( k )th fund seeker</td>
</tr>
<tr>
<td>Let ( tp ) be the total projects handled by ( k )th fund seeker</td>
</tr>
<tr>
<td>prob(_{tpc}) = ( tpc/tp )</td>
</tr>
<tr>
<td>prob(_{tpd}) = ( tpd/tp )</td>
</tr>
<tr>
<td>prob(<em>{either}) = ( (prob(</em>{tpc}) + prob(<em>{tpd})) - (prob(</em>{tpc}) * prob(_{tpd})) )</td>
</tr>
<tr>
<td>EVAL(( i, j, k )) = prob(_{either})</td>
</tr>
<tr>
<td>Else if (( j = 6 )) then</td>
</tr>
<tr>
<td>Let ( tb ) be total budget proposed by ( k )th fund seeker</td>
</tr>
<tr>
<td>Let ( tr ) be total budget required by ( k )th fund seeker</td>
</tr>
<tr>
<td>EVAL(( i, j, k )) = ( tr/tb )</td>
</tr>
<tr>
<td>Else if (( j = 7 )) then</td>
</tr>
<tr>
<td>EVAL(( i, j, k )) = npv</td>
</tr>
<tr>
<td>Else if (( j = 8 )) then</td>
</tr>
<tr>
<td>Let ( s ) be sum of infrastructure available (infrastructure ID wise) for all the qualified fund seeker provided seeker type is individual otherwise 1 to all.</td>
</tr>
<tr>
<td>EVAL(( i, j, k )) = infrastructure available for ( k )th fund seeker</td>
</tr>
<tr>
<td>Else if (( j = 9 )) then</td>
</tr>
<tr>
<td>Find total number management capability keywords in ( k )th proposal and let it be ( kw )</td>
</tr>
<tr>
<td>EVAL(( i, j, k )) = ( kw )</td>
</tr>
<tr>
<td>Else if (( j = 10 )) then</td>
</tr>
<tr>
<td>Let ( s ) be sum of manpower available (Designation wise) for all the qualified fund seeker staff(( i )) = manpower available for ( k )th fund seeker</td>
</tr>
<tr>
<td>EVAL(( i, j, k )) = 1 - EVAL(( i, 6, k ))</td>
</tr>
<tr>
<td>Else if (( j = 11 )) then</td>
</tr>
<tr>
<td>EVAL(( i, j, k )) = Average of Risk Analysis factors' value for ( k )th Proposals</td>
</tr>
</tbody>
</table>
| End if End

Table 4.16: Pseudo Code for Evaluating Proposals

Step V of the algorithm has three sub-tasks to perform. Each numeric value calculated in step IV is multiplied by weight. The weighted values are further clubbed into
single value by getting GM. The value is concerned with individual proposals. Table 4.17 describes the steps using pseudo code. Step V(b) of the algorithm calculates the three cut offs based on percentile policy set by the fund allocators. Against each percentile policy, there is a limit of maximum percentage of the funds demanded that can be given to fund seeker. Proposals falling below the last cut off are denied funds. Steps have been explained in table 4.18 using pseudo code. Last step is to distribute funds according to rank of the proposal, availability of funds, maximum fund under category in consideration and cut offs. Step V (c) is detailed in table 4.19.

### Allocating Funds (Step V-(a))

```plaintext
For i = 1 to n
    begin
        If (PROJECT_STATUS(i) = 'QUALIFIED'
            For j = 1 to 12
                begin
                    Multiple calculated value by by EW,
                End for
            Calculate Geometric Mean
        End if
    End for
```

Table 4.17: Weight-based Algorithm (Step V-(a))

### Calculate Cutoffs (Step V-(b))

```plaintext
Let s be the sorted array of fund seeker according to weight in ascending order whose project status is 'Qualified' for lth Category
P_l = (perl * (r+1))/100
If p_l is whole number and p_l > r then
    P_l = p_l; high = r;
Else If p_l is whole number and p_l < 1 then
    P_l = p_l; high = 1
Else If p_l is whole number and p_l > 1 then
    P_l = p_l; low = p_l; high = p_l
End if
If P_l is not whole number then
    P_l = floor(p_l)
    P_l = ceil(p_l)
End if
If P_l < 1 then
    P_l = 1
If P_l > r then
    P_l = r
P_l = s(p_l; low)
P_l = s(p_l; high)
Cutoff1 = (p_l; low + p_l; high)/2
Similarly calculate cutoff2 and cutoff3
```

Table 4.18: Weight-based Algorithm (Step V-(b))

The agents of the system perform actions at the different time and at the different locations. The steps mentioned in the previous section cover all the actions performed by agents at the different times and locations. Hence, it becomes important that algorithm should be distributed one. Table 4.20 shows the distribution of tasks of algorithm among agents. Agents can also work in parallel, e.g. if FSA is busy in gathering information from fund seeker for the fund category FC1, FAMA may be busy in evaluating and allocating the
budget to fund seekers from fund category FC2 and FC3. FSA passes the information to FAMA using database and well-defined ontology. Distributed Algorithm completes the task when all participating agents complete their tasks. Step V of the algorithm is further divided into multiple tasks and takes advantage of multi threading. There are 12 different decision-making factors categorized into five high level factors. These are independent except C5.1 as it is dependent on C2.2. The computations of these factors are multithreaded taking advantage of both parallel and distributed environment. The distributed algorithm is described in Table 4.21 and 4.22.

**Distributing Funds (Step V-(c))**

For each category

Let cl array, a array of size n1 contains lists of fund seekers having weights greater than or equal to cutoff1 and diarray contains demanded funds.

\[ \text{Sum of demand} = \text{sum of (elements of clarray)} \times \text{Max1/100} \]

If (sum_of_demand < fund availability then

Allocate \( \min(\text{diarray}, \times \text{Max1/100}, \text{maxcap1}) \) to clarray where \( i \geq 1 \) and \( \leq n1 \)

Else Allocate available funds proportionately to clarray subject to \( \min(\text{diarray}, \times \text{Max1/100}, \text{maxcap1}) \)

End if

Allocate allocated fund to AR(Li) (ith category and ith proposal).
Calculate remaining funds
Repeat above for second and third cut offs.
Deny funds to fund seekers having weight less than cut off 3.

**Table 4.19: Weight-based Algorithm (Step V-(c))**

<table>
<thead>
<tr>
<th>Agent</th>
<th>Steps of Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSA</td>
<td>Step I</td>
</tr>
<tr>
<td>FAMA</td>
<td>Step I, II, III, IV, and V</td>
</tr>
<tr>
<td>FCA</td>
<td>To interact with user through web based interface to set parameters</td>
</tr>
</tbody>
</table>

**Table 4.20: Agent wise Tasks Allocation**

// Distributed Algorithm
Begin
Start thread FS_thread to take input from the fund seeker user as a part of Step I.
Start thread FA_thread to take input from the fund allocator user as a part of Step I.
Start thread EXP_thread to take input from the expert user as a part of Step I.
Do while fund requirement found and budget found
Begin //assign task to Fund Allocator and Monitor Agent
Do Step I
For \( i = 1 \) to \( m \) // allocating categories
Create thread \( t_i \) // calls budget procedure
End for
Thread_wait //Wait for all the active threads to finish
End

**Table 4.21: Distributed Algorithm**

75
Chapter 4. Design of Multi-agent System for Resource Allocation and Monitoring

4.4. Supporting Procedures

In designing supporting procedures, procedures required to perform actions are detailed and specified using pseudo code. Table 4.23 shows the actions performed by three agents of MASRAM.

<table>
<thead>
<tr>
<th>Agent</th>
<th>Action</th>
</tr>
</thead>
</table>
| FSA   | 1. Profile - FSU (Fund Seeker User)  
|       | 2. Authentication - FSU  
|       | 3. Finding funding Sources  
|       | 4. Proposal Submission  
|       | 5. Informing utilization status |
| FAMA  | 1. Authentication - FAU (Fund Allocator User)  
|       | 2. Allocation of funds  
|       | 3. Monitoring Progress  
|       | 4. Allocation status  
|       | 5. Reviewing allocation  
|       | 6. Reviewed allocation  
|       | 7. Profile - FAU |
| FCA   | 1. Preparing contents and calling agent FSA on request of fund seeker.  
|       | 2. Preparing contents and calling agent FAMA on request of fund allocator or reviewer.  
|       | 3. Informing users the response received from agent FSA/FAMA |

Table 4.23: Agent wise Actions Performed

4.4.1. Actions Performed by FSA

Procedures for actions performed by FSA are given below.

4.4.1.1. Procedure Profile - FSU

This procedure has been designed to validate the particulars given by FSU to create login. The procedure after validating input, stores data in database and creates unique seeker identification number. The procedure returns status of validation and seeker identification number. Table 4.24 describes the pseudo code.

4.4.1.2. Procedure Validate_FSU

This procedure takes login ID and password of FSU and validates the credential with the data stored in database. It returns the status (success/fail) as shown in table 4.25.
4.4.1.3. Procedure find_sources

This procedure finds the sources of funding agencies based on nature of project, budget required and type of fund seeker and returns the same. The procedure is described in table 4.26.

```plaintext
para operation_code, seeker_name, seeker_address, seeker_phone, seeker_legal_status, seeker_experience, seeker_email, seeker_pwd, seeker_confirm_pwd, seeker_profile, error_msg out, status out, seeker_id in out
Begin
  status = '1'
  if seeker_pwd is not equal to seeker_confirm_pwd then
    error_msg = 'Password and confirm password do not match'
    status = 'Fail'
  End if
  if seeker_name contains numerics or seeker_name is empty or length is greater than 100 characters then
    error_msg = 'Name should contain maximum 100 alphabets'
    status = 'Fail'
  End if
  if seeker_email does not contain single '(@)' and one or more dots then
    error_msg = 'Invalid email address'
    status = 'Fail'
  End if
  if seeker_experience is not greater than zero then
    error_msg = 'Invalid experience'
    status = 'Fail'
  End if
  if seeker_phone contains alphabet or special character or length > 11 or negative then
    error_msg = 'Invalid phone'
    status = 'Fail'
  End if
  if status = '1' then
    if operation_code = 'insert' then
      Insert data into seeker_mas and generate seeker_id
    Else
      Update seeker_mas based on seeker_id
    End if
    status = 'success'
  End if
  Exception
    status='fail'
  Return
End
```

Table 4.24: Pseudo Code for Profile of FSU

```plaintext
Para login_id, pwd, status out
begin
  status = 'fail'
  find password from seeker_mas database based on login_id
  if found them
    compare passwod with pwd
    if matches them
      status = 'success'
    end if
  end if
End
```

Table 4.25: Pseudo Code for Validate FSU
Para nature_of_project, list out, status out, fund_range, type_of_seeker

Begin
  Status = 'O'
  prepare resultset from fund_category table
  while next records found from resultset
    if nature_of_project from resultset matches with nature_of_project,
      budget required and type of fund seeker then
      add records to list
      status = '1'
    end if
  end while
End

Table 4.26: Pseudo Code for Finding Sources

4.4.1.4. Procedure proposal_submission

This procedure takes input related with proposal and after validating input, stores proposals in the relevant tables and nested tables. If data is stored successfully, it generates project identification number (Project_ID) and returns the same. The pseudo code is written in table 4.27.

4.4.1.5. Procedure utiljstatus

This procedure takes the progress and utilization of funds related with project, validates data and stores in relevant table. On successful storage, it sends '0' and on failure due to any reason, it returns '1' as shown in table 4.28.

Para object in_proposal_jnain, object in_schedule, object in_budget_required, object ininfra, object in_manpower, object in_risk_analysis, object in_methodology, object in_tech_used, object in_project_done, object in_outcome, in_out_project_id out, out_status out, object in_cost_analysis, object benefit_analysis

Begin
  Out_status = 'O'
  If in_out_project_id is null then
    Insert in_proposal_main in project_proposals table.
    Find newjprojectid
    Inoutjprojectid = new_projec_id
    Out_status='1'
  End if
  If in_schedule is not null then
    Update project_proposals with object in_schedule
    Out_status='1'
  End if
  If in_budget_required is not null then
    Update project_proposals with object in_budget_required
    Out_status='1'
  End if
End

Table 4.27: Pseudo Code for Proposal Submission (Contd..)
Chapter 4. Design of Multi-agent System for Resource Allocation and Monitoring

If in infra is not null then
    Update project_proposals with object in infra
    Out_status='1'
End if
If in manpower is not null then
    Update project_proposals with object in manpower
    Out_status='1'
End if
If in risk_analysis is not null then
    Update project_proposals with object in risk_analysis
    Out_status='1'
End if
If in methodology is not null then
    Update project_proposals with object in methodology
    Out_status='1'
End if
If in tech_used is not null then
    Update project_proposals with object in tech_used
    Out_status='1'
End if
If in project_done is not null then
    Update project_proposals with object in project_done
    Out_status='1'
End if
If in outcome is not null then
    Update project_proposals with object in outcome
    Out_status='1'
End if
Insert object_cost_analysis and object_benefit_analysis in project_cba
End

Table 4.27: Pseudo Code for Proposal Submission

Para in_project_id, in_activity_id, in_util_date, in_util_funds, in_remarks, out_status out
Begin
    Out_status = '1'
    Update nested table milestone_nt of table project_proposals where in_project_id matches with
    project_id and in_activity_id with activity_id
    Out_status = '0'
    Exception
    Out_status = '1'
End

Table 4.28: Pseudo Code for Progress Updation

4.4.2. Actions Performed by Fund Allocator and Monitor Agent (FAMA)

FAMA assists fund allocator user by performing actions. It takes data from FCA/Database and performs the required tasks. It stores the profiles of users. It also reviews the allocation made by system. Following pseudo codes show the tasks performed by FAMA.

79
4.4.2.1. Procedure validate_user

This procedure takes input (login, password and type of user) and checks with database. It checks login credentials of allocator, expert and reviewer users. It returns the status in terms of successful (0) or failure (1). Table 4.29 shows the pseudo code of this procedure.

```
Para in_user_type, in_login_id, in_pwd, out_status out
Begin
    Out_status = '1'
    If in_user_type is 'reviewer' then
        Find pwd from reviewer_mas where reviewer_id matches with in_login_id
        If found then
            If pwd matches with in_pwd then
                Out_status = '0'
            End if
        End if
    Else
        If in_user_type is 'allocator' then
            Find pwd from allocator_mas where allocator_id matches with in_login_id
            If found then
                If pwd matches with in_pwd then
                    Out_status = '0'
                End if
            End if
        Else
            If in_user_type is 'expert' then
                Find pwd from experts_mas where expert_id matches with in_login_id
                If found then
                    If pwd matches with in_pwd then
                        Out_status = '0'
                    End if
                End if
            Else
               Exception Out_status = '1'
            End if
        End if
End if
```

Table 4.29: Pseudo Code for Validating User

4.4.2.2. Procedure monitoring_progress

This procedure finds the proposals that are sanctioned grants and are due to submit their progress but have not timely submitted the progress and utilization of funds. It prepares reminders for such proposals and sends to FSA. This procedure also decides the action to be taken based on mid-term report or final report. If the progress is due for mid-term report, based on feedback by experts, the decision is taken either to continue with project or to terminate. If final report is due for particular project, action could be to know successful or unsuccessful status of project. The procedure considers fund seeker’s progress report if the feedback from experts is not required. Table 4.30 describes all the steps followed in this procedure.
Chapter 4. Design of Multi-agent System for Resource Allocation and Monitoring

Begin
Prepare cursor/resultset cl from milestone nt, nested table of mail table project_proposals where ending date of the records is less than current date and status is not received
Open cl
While cl has records
  Prepare reminder and communicates to FSA through database
End while
Prepare cursor cl for all the projects whose progress reports are due
For every project in cl
    Let no_of_feedback be the number of feedbacks required from experts
    Let no_of_recd_feedback be the number of feedbacks received from experts
    Let seeker_rating be overall rating given by seeker
    If no_of_feedback = 0 then
        If seeker_rating is not null then
            Overall_rating = seeker_rating
        Else
            Status = 'pending'
        End if
    Else
        If no_of_recd_feedback > 0 then
            let Overall_rating be average rating of all the feedbacks received
            else
                status = 'pending'
            end if
        end if
    end if
    if status <> 'pending' then
        if overall rating is at least 'Good' and project is in progress then
            status = 'To be continued'
            allocate further funds if required
        else
            if overall rating is below 'Good' and project is in progress then
                status = 'Stop further funds'
            else
                if overall rating at least 'Good' and project is completed then
                    status = 'Successful'
                else
                    status = 'Unsuccessful'
                end if
            end if
        end if
    end if
End for
End

Table 4.30: Pseudo Code for Monitoring Progress
4.4.2.3. Procedure allocation_status

This procedure prepares the list of proposals that are allocated funds on particular date. There may be more than one allotment against single fund category, so batch number or lot number is must to get the result. Table 4.31 shows the pseudo code of allocation.

| Para in_seeker_id, in_fund_category, in_start_date, in_end_date, in_project_id ,in_batchno , out_list out |
| Begin // one of the following if block will be executed at a time. |
| If in_seeker_id is not null then |
| Prepare cursor/resultset c1 from allocation, project_proposals tables where seeker_id matches with in_seeker_id and project ed of allocation matches with project ed of project_proposals and in_batchno matches with batchno of allocation. |
| Open c1 |
| While c1 has records |
| Add to out_list |
| End while |
| End if |
| If in_fund_category is not null then |
| Prepare cursor/resultset c1 from allocation, project_proposals tables where nature_of_project of fund_category object matches with in_fund_category and project_id of project proposals matches with project_id of allocation. |
| Open c1 |
| While c1 has records |
| Add to out_list |
| End while |
| End if |
| If in_start_date is not null then |
| Prepare cursor/resultset c1 from allocation, project_proposals tables where rank_date is between in_start_date and in_end_date and project_id of allocation matches with project_id of project_proposals |
| Open c1 |
| While c1 has records |
| Add to out_list |
| End while |
| End if |
| If in_project_id is not null then |
| Prepare cursor/resultset c1 from allocation, project_proposals tables where project_id of allocation matches with in_project_id and project_id of project_proposals |
| Open c1 |
| While c1 has records |
| Add to out_list |
| End while |
| End if |
| End |

Table 4.31: Pseudo Code for Allocation Status
4.4.2.4. Procedure reviewing_allocation

This procedure prepares the list of proposals that are allocated funds by FAMA corresponding to particular Reviewer ID and returns the prepared list of projects for reviews. The pseudo code is given in table 4.32.

```
Para in_review_id, out_list out
Begin
  Prepare res|ultset/cursor cl from allocation, project_proposals and reviewer_allocator_mas tables where project_id of allocation matches with project_id of project_proposals and reviewer_id of reviewer_allocator_mas matches with in_review_id
  Open cl
  While cl has records
    Add to out_list
  End while
End
```

Table 4.32: Pseudo Code for Reviewing Allocation

4.4.2.5. Procedure reviewed_allocation

Once reviewer reviews the proposals, the final allocation is made by calling this procedure. The pseudo code is given in table 4.33.

```
Para in_review_id, object in_allocation_object
Begin
  Prepare resultset/cursor cl from reviewer_allocator_mas, allocation where reviewer_id matches with in_review_id and project_id of in_allocation_object matches with project_id of allocation
  Update allocation based on record cl with in_allocation_object
  Update project_proposals change status to 'Finally Allocated' where project_id matches with project_id of in_allocation_object
End
```

Table 4.33: Pseudo Code for Reviewed Allocation

4.4.2.6. Procedure profile_FAU

This procedure, after verifying data entered by fund allocator user and reviewer user, inserts data into appropriate table. Once data is saved successfully, unique identification number is generated and returned. In case of failure, error message code is returned. The steps are given in table 4.34.
4.4.3. Actions Performed by Facilitator Agent (FCA)

The aim of FCA is to provide interfaces to users so that users can provide required information to agents of the system for the purpose of decision-making of funds allocation and monitoring.

```
Para in_allocator_id, in_allocator_name, in_contact_person, in_allocator_address, in_allocator_phone, in_allocator_email, in_allocator_pwd, in_reviewer_id, in_reviewer_name, in_reviewer_address, in_reviewer_phone, in_reviewer_email, in_reviewer_pwd, in_user_type, out_status out
Begin
If in_user_type = 'allocator' then
  Validate all fields related with allocator as per database
  If error in validation
    Out_status = '1-' + invalid field
    Return
  Else
    Insert data into allocator_mas
    Out_status = '0'
    Return
  End if
Else
  If in_user_type = 'reviewer' then
    Validate all fields related with reviewer as per database
    If error in validation
      Out_status = '1-' + invalid field
      Return
    Else
      Insert data into reviewer_mas
      Out_status = '0'
      return
    End if
  End if
End if

Table 4.34: Pseudo Code for Profile – FAU
```

The detailed design of the system helped in implementing agents of MASRAM using JADE (Java Agent Development Framework), interfaces using JSP (Java Server Pages), and communication through use of ontology and Oracle as database at backend. Next chapter describes the implementation detail of MASRAM.