Ethan Cox
3/21/18
TIM 58

Final

1. Requirements definition

Nonfunctional Requirements

1. Operational Requirements
    - 1.1 System can operate on any web browser.
    - 1.2 System will automatically back up at midnight.
    - 1.3 System must operate in desktop environment.
2. Performance Requirements
    - 2.1 System must be available 24 hours a day, 365 days a year
    - 2.2 Response time between system and the user must be 5 seconds or less
    - 2.3 System will store and retrieve transactional information every three seconds.
3. Security requirements
    - 3.1 Information about customers issues limited to Program Officers and the Expert Panel.
4. Cultural and political requirements
    - 4.1 The system must comply with regulations regarding customer privacy.
    - 4.2 System can handle any language.

Functional Requirements

1. Track grants through all of their stages and maintain records of whether they have been completed successfully or failed.
    - 1.1 Board of Directors generates a list of grants for the Environmental Conservation Program. Each grant is given a name, key words (e.g., coral reefs, marine mammals), and dollar value, and is assigned to a single Program Officer.
    - 1.2 Program Officer writes the Request for Proposal (RFP) for each grant they receive.
    - 1.3 Select experts who have the same keyworks as the grant.
    - 1.4 Request for Proposal sent to potential research teams.
        - 1.4.1 If no proposal is submitted mark grant failed.
        - 1.4.2 If proposals submitted, they're forwarded to Expert Panel.
    - 1.5 Expert Panel selects best proposal and winning research team.
    - 1.6 Winning research team conducts research and submits final report to program officer.
    - 1.7 Program Officer sends final report to Expert Panel for review.
        - 1.7.1 If report needs changes it is sent back to review who makes recommended changes. If Expert Panel is satisfied with changes, grant is marked complete.
        - 1.7.2 If report doesn't need changes the grant is marked complete.
        - 1.7.3 If report not submitted grant is marked failed.

2. Help Program Officers keep track of their grants through all stages of the grant

Process.

2.1 Desktop environment shows all grants submitted by Board of Directors.

2.2 Desktop environment lets PO create new Expert Panel and add experts to it.

2.3 Desktop environment lets PO add RFP to RFP database and shows all active RFP’s.

2.4 Desktop environment lets PO see each experts winning proposal for any grant.

2.5 Desktop environment lets PO check if winning research team has failed any grants in the past.

2.6 Desktop environment lets PO submit final report to Experts.

2.7 Desktop environment lets PO send reports to research team.

3. Maintain detailed database records on each grant, on experts, and on Research Teams.

3.1 Foundation manages database of grants, and is able to create new grants and mark them complete or failed.

3.2 Foundation manages database of experts, and is able to update skills for any expert and track what RFP’s they are working on.

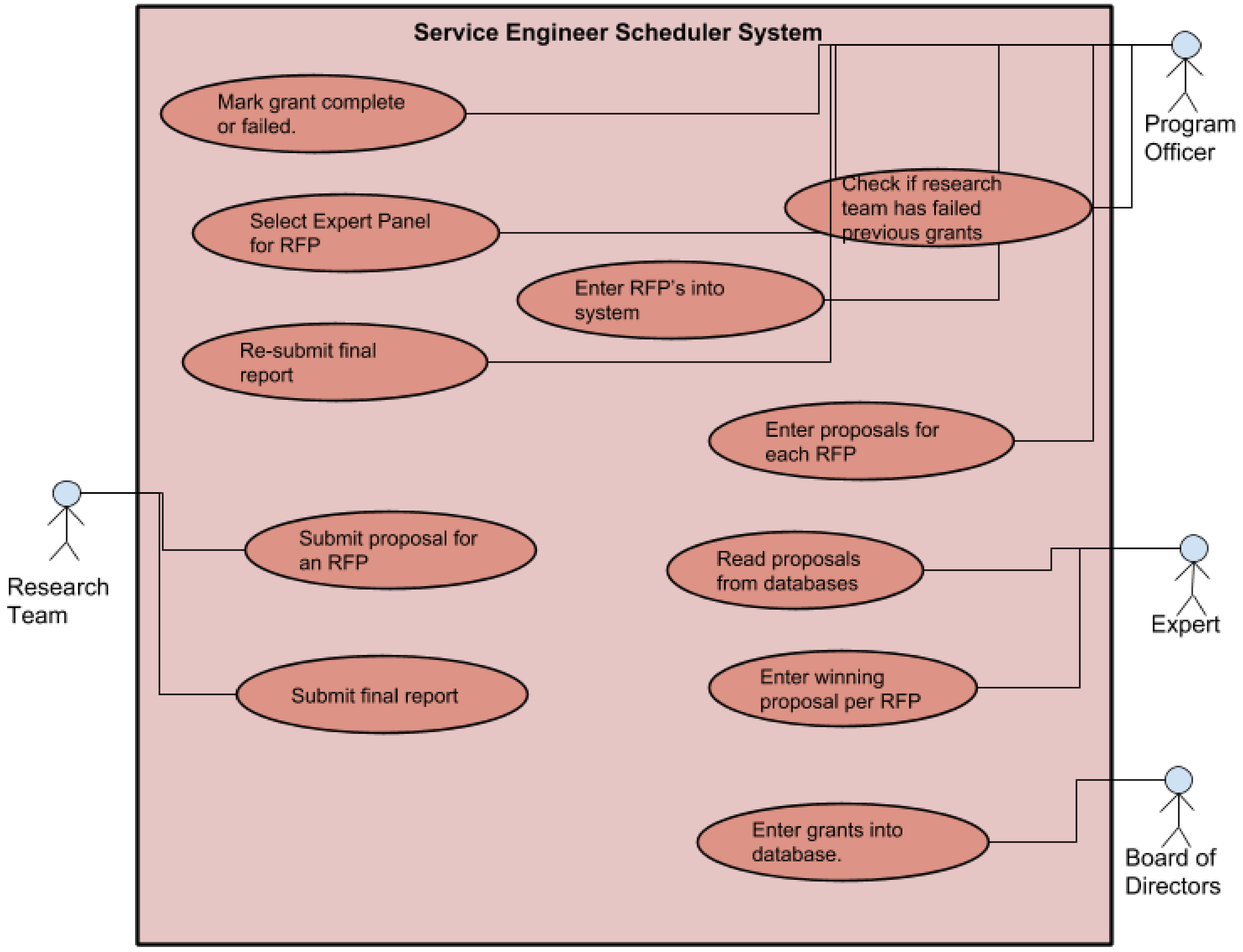
3.3 Foundation manages databases of research teams, including any grants they worked on in the past and the result of those grants and RFP’s they’re currently working on.

3.4 Foundation maintains a database of final reports from previous grants.

2. Use-case diagram

Use Cases:

1. Board of Directors enter initial grants for the year into the database.
2. PO creates RFP’s for each grant they are given, enters RFP’s into database.
3. PO uses system to create new expert panel and select experts for the panel.
4. Mark grant failed if no proposals submitted by deadline.
5. Put proposals into database to be reviewed by expert panel.
6. Experts read proposal from database, select winning proposal and mark winning proposal in database.
7. PO checks database to see if winning team has failed grants in past.
8. After research is completed, PO puts Final Report in system.
9. Experts read final report, put comments in system.
10. PO resubmits final report.


Service Engineer Scheduler System
Mark grant complete or failed.
Select Expert Panel for RFP
Check if research team has failed previous grants
Enter RFP's into system
Re-submit final report
Enter proposals for each RFP
Submit proposal for an RFP
Read proposals from databases
Submit final report
Enter winning proposal per RFP
Enter grants into database.
Program Officer
Research Team
Expert
Board of Directors

3. Activity diagram

1. Track grants through all of their stages and maintain records of whether they have been completed successfully or failed.

    1.1 Board of Directors generates a list of grants for the Environmental Conservation Program. Each grant is given a name, key words (e.g., coral reefs, marine mammals), and dollar value, and is assigned to a single Program Officer.

    1.2 Program Officer writes the Request for Proposal (RFP) for each grant they receive.

    1.3 Select experts who have the same keywords as the grant.

    1.4 Request for Proposal sent to potential research teams.

        1.4.1 If no proposal is submitted mark grant failed.

        1.4.2 If proposals submitted, they're forwarded to Expert Panel.

    1.5 Expert Panel selects best proposal and winning research team.

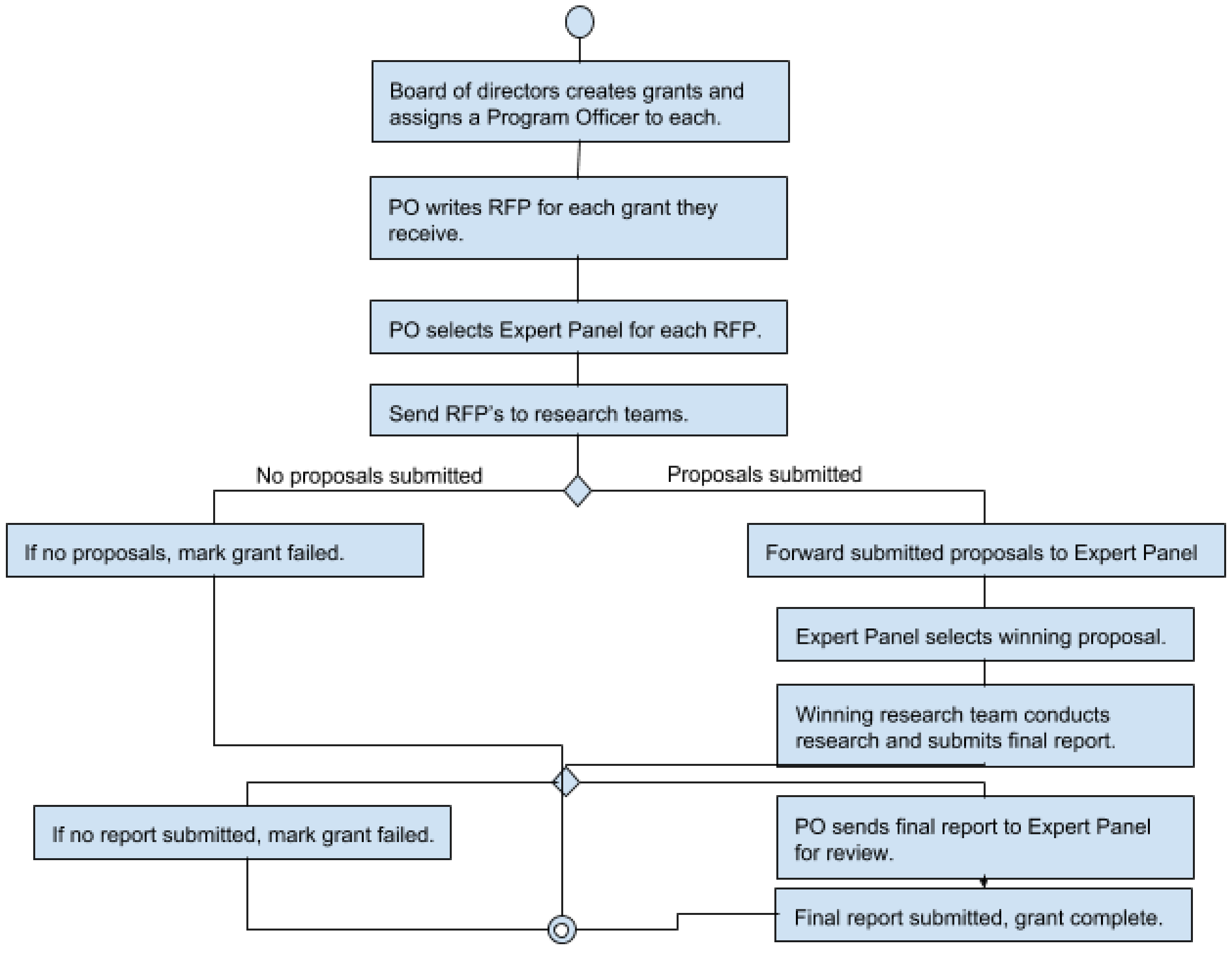
1.6 Winning research team conducts research and submits final report to program officer.

1.7 Program Officer sends final report to Expert Panel for review.

1.7.1 If report needs changes it is sent back to review who makes recommended changes. If Expert Panel is satisfied with changes, grant is marked complete.

1.7.2 If report doesn't need changes the grant is marked complete.

1.7.3 If report not submitted grant is marked failed.


Board of directors creates grants and assigns a Program Officer to each.
PO writes RFP for each grant they receive.
PO selects Expert Panel for each RFP.
Send RFP's to research teams.
No proposals submitted
Proposals submitted
If no proposals, mark grant failed.
Forward submitted proposals to Expert Panel
Expert Panel selects winning proposal.
Winning research team conducts research and submits final report.
If no report submitted, mark grant failed.
PO sends final report to Expert Panel for review.
Final report submitted, grant complete.

2. Help Program Officers keep track of their grants through all stages of the grant Process.

2.1 Desktop environment shows all grants submitted by Board of Directors.

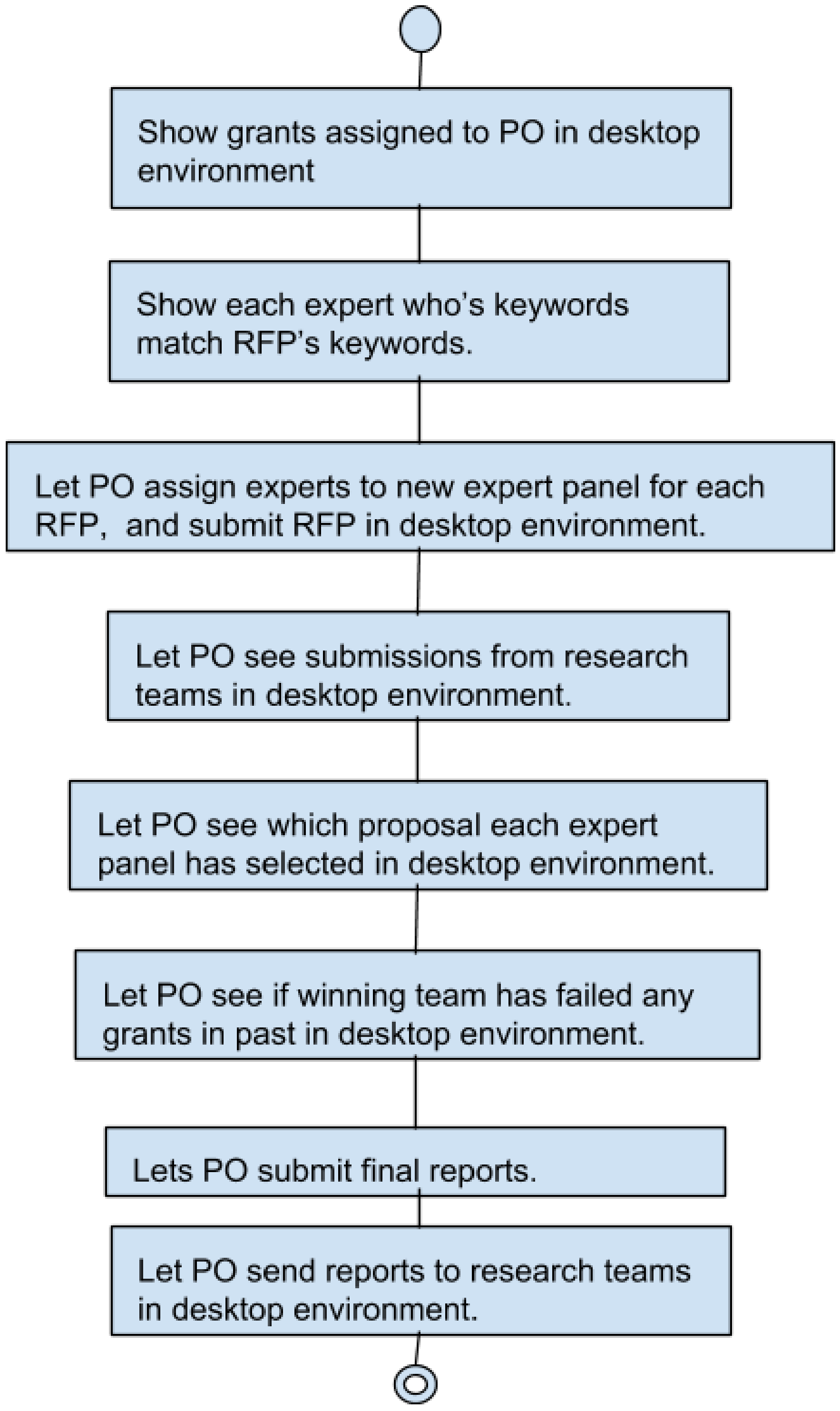
2.2 Desktop environment lets PO create new Expert Panel and add experts to it.

2.3 Desktop environment lets PO add RFP to RFP database and shows all active RFP's.

2.4 Desktop environment lets PO see each experts winning proposal for any grant.

2.5 Desktop environment lets PO check if winning research team has failed any grants in the past.

2.6 Desktop environment lets PO submit final report to Experts.
2.7 Desktop environment lets PO send reports to research team.


Show grants assigned to PO in desktop environment
Show each expert who's keywords match RFP's keywords.
Let PO assign experts to new expert panel for each RFP, and submit RFP in desktop environment.
Let PO see submissions from research teams in desktop environment.
Let PO see which proposal each expert panel has selected in desktop environment.
Let PO see if winning team has failed any grants in past in desktop environment.
Lets PO submit final reports.
Let PO send reports to research teams in desktop environment.

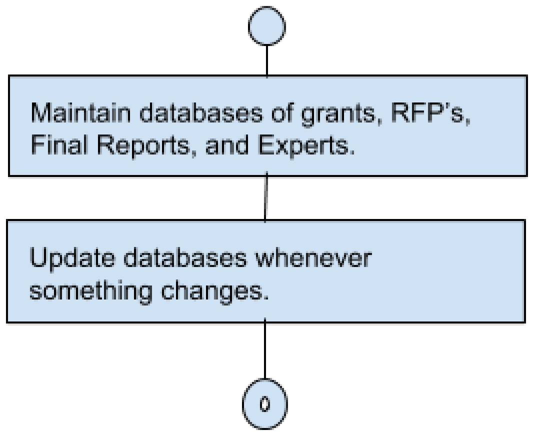
3. Maintain detailed database records on each grant, on experts, and on Research Teams.

3.1 Foundation manages database of grants, and is able to create new grants and mark them complete or failed.

3.2 Foundation manages database of experts, and is able to update skills for any expert and track what RFP's they are working on.

3.3 Foundation manages databases of research teams, including any grants they worked on in the past and the result of those grants and RFP's they're currently working on.

3.4 Foundation maintains a database of final reports from previous grants.


Maintain databases of grants, RFP's, Final Reports, and Experts.
Update databases whenever something changes.
0

4. Use-case descriptions

Use Case	ID:1 Grants created and entered in database
Actors	Board of Directors
Stakeholders	BD - wants to enter grants into database
Description	The BD enters created grants in the database.
Trigger	The new year starts, it's time to create a new set of grants.
Relationships	Association: Board of Directors, System
Flow of Events	1. Create grants for the new year. 2. Enter them in the Grants db.
Alternate flows	

Use Case	ID:2 RFP entered into database
Actors	Program Officer
Stakeholders	Program Officer - wants to enter RFP in database so it can be seen by Experts
Description	The Program Officer enters RFP's into database for each grant they are assigned.
Trigger	Program Officer receives grants, creates RFP's for each report, enters them in the system.

Relationships	Association: Program Officer, System
Flow of Events	3. Receive grants, create RFP’s for each grant. 4. Submit RFP’s for system.
Alternate flows	- Remove RFP from system.

Use Case	ID:3 Create new expert panel for each RFP and assign experts to the panel.
Actors	Program Officer
Stakeholders	Program Officer - wants to create expert panel for each RFP.
Description	The Program Officer creates an expert panel for an RFP.
Trigger	Program Officer has created and submitted RFP, creates expert panel.
Relationships	Association: Program Officer, Experts
Flow of Events	1. Submit RFP 2. Create new expert panel and assign it to RFP
Alternate flows	

Use Case	ID:4 Update grant status
Actors	Program Officer
Stakeholders	Program Officer - needs to update status of grant
Description	PO changes status of grant depending on what happened with it.
Trigger	Proposals not sent to RFP, report not created, or final report accepted by Expert Panel.
Relationships	Association: Program Officer, Research Team, System
Flow of Events	1. Proposal not sent, report not created, or final report accepted by Expert Panel. 2. Mark proposal failed or complete.
Alternate flows	

Use Case	ID:5 Proposals placed into database.
Actors	Program Officer
Stakeholders	Program Officer - wants to put submitted proposals into database so experts can read them.
Description	The Program Officer submits the proposals into the system so the experts can read them.
Trigger	Program Officer receives proposals from research teams.

Relationships	Association: Program Officer, System
Flow of Events	1. Proposals sent in from research teams for each RFP. 2. Proposals logged in database
Alternate flows	- Remove proposals from system.

Use Case	ID:6 Experts select winning proposal and log it in system
Actors	Expert
Stakeholders	Expert - wants to select proposal they believe is best. Program Officer - wants to know who has best proposal so they can alert research team. Research team - need to know who won grant.
Description	The expert reads each proposal for a RFP they have been assigned to and select the best one, logging the choice in the system.
Trigger	Expert receives proposals for RFP they have been assigned to work on.
Relationships	Association: Expert, System
Flow of Events	1. Expert receives proposals for RFP they have been assigned to work on. 2. Expert reads proposals and selects their winner. 3. Expert logs winner in system.
Alternate flows	

Use Case	ID:7 PO checks database to see if winning team has failed grants in past.
Actors	Program Officer
Stakeholders	Program Officer - wants to know if selected research team has failed any grants in the past. Research team - needs to know if they will be able to work on grant.
Description	The PO checks database of grants to see if research team selected by expert panel has failed any grants they worked on in the past.
Trigger	Program officer is alerted of select proposal sent by research team for a RFP.
Relationships	Association: Program Officer, System
Flow of Events	1. Program Officer alerted that expert panel has selected a winning proposal for an RFP. 2. Program Officer checks if winning research team has failed any grants before. 3. If so, PO asks expert panel to select a different proposal.
Alternate flows	

Use Case	ID:8 PO submits initial copy of final report.
Actors	Program Officer
Stakeholders	Program Officer - wants to know if final report is satisfactory. Expert Panel - wants to check if final report is satisfactory.
Description	The PO submits the final report into the database.
Trigger	Program Officer receives copy of final report from research team.
Relationships	Association: Program Officer, System
Flow of Events	1. Program Officer receives copy of final report from research team. 2. PO submits report into system (Final Reports db).
Alternate flows	

Use Case	ID:9 Experts log comments about copy of final report into system.
Actors	Experts
Stakeholders	Experts - want to alert research team about changes that need to be made for final report to be acceptable. Program Officer - wants to know if grant is complete. Research team - want to know if final report is acceptable.
Description	Experts read the final report submitted by the PO and respond with comments about the issues they see with the current state of the report.
Trigger	Experts are alerted that a new copy of the final report for a RFP has entered into the system.
Relationships	Association: Expert, System
Flow of Events	1. Experts are alerted that a new copy of the final report for a RFP has entered into the system. 2. Experts log comments about copy of final report into system.
Alternate flows	- Experts approve final report, grant is marked complete.

5. CRC cards

The classes for the system are:

1. Employee
2. Board of Directors member
3. Program Officer
4. Expert
5. Research Team member

The CRC cards for the classes are:

Front:

Class name: Employee	ID:1	Type: Concrete, Domain
Description: Employees work for the foundation, this is a super class.		Associated use cases: 1,2,3,4,5,6,7,8
Responsibilities: All functionality of the system.		Collaborators: Program Officers Board of Directors members Experts Research Teams

Back:

Attributes:

Name (string)
Role (string)

Relationships:

Generalization: Employee

Aggregation: Role

Other associations: none

Front:

Class name: Board of Directors Member	ID:2	Type: Concrete, Domain
Description: Board of Directors are responsible for determining the set of grants for the year.		Associated use cases: 1
Responsibilities: Determine the set of grants for the year.		Collaborators: Program Officers Experts Research Teams

Back:

Attributes:

Name (string)
Role(string)
Expertise (string)

Relationships:

Generalization: Employee

Aggregation: none

Other associations: none

Front:

Class name: Program Officer	ID:3	Type: Concrete, Domain
Description: Program Officers are responsible for ensuring each grant is fulfilled.		Associated use cases: 2,3,4,5,7,8

Responsibilities:	Collaborators:
Create RFPs	Program Officers
Select Expert Panels	Experts
Request proposals for RFPs	Research Teams
Send proposals to Experts	
Submit final reports	

Back:

Attributes:

Name (string)
Role(string)
Projects Managed (array[int])

Relationships:

Generalization: Employee

Aggregation: projects (they're managing)

Other associations: none

Front:

Class name: Expert	ID:4	Type: Concrete, Domain
Description: Experts are responsible for selecting the best proposal for each grant.		Associated use cases: 5,9
Responsibilities: Selecting the best proposal for each grant. Review final reports.		Collaborators: Program Officers Bo Research Teams

Back:

Attributes:

Name (string)
Role (string)
Keywords (string)

Relationships:

Generalization: Employee

Aggregation: keywords (i.e. areas of expertise)

Other associations: none

Front:

Class name: Research Team member	ID:5	Type: Concrete, Domain
Description: Research Team members are responsible for fulfilling the grant.		Associated use cases:
Responsibilities: Submit proposals for grants. Perform research. Submit final report.		Collaborators: Program Officers Board of Directors Experts

Back:

Attributes:

Name (string)
Role (string)
Project (int)

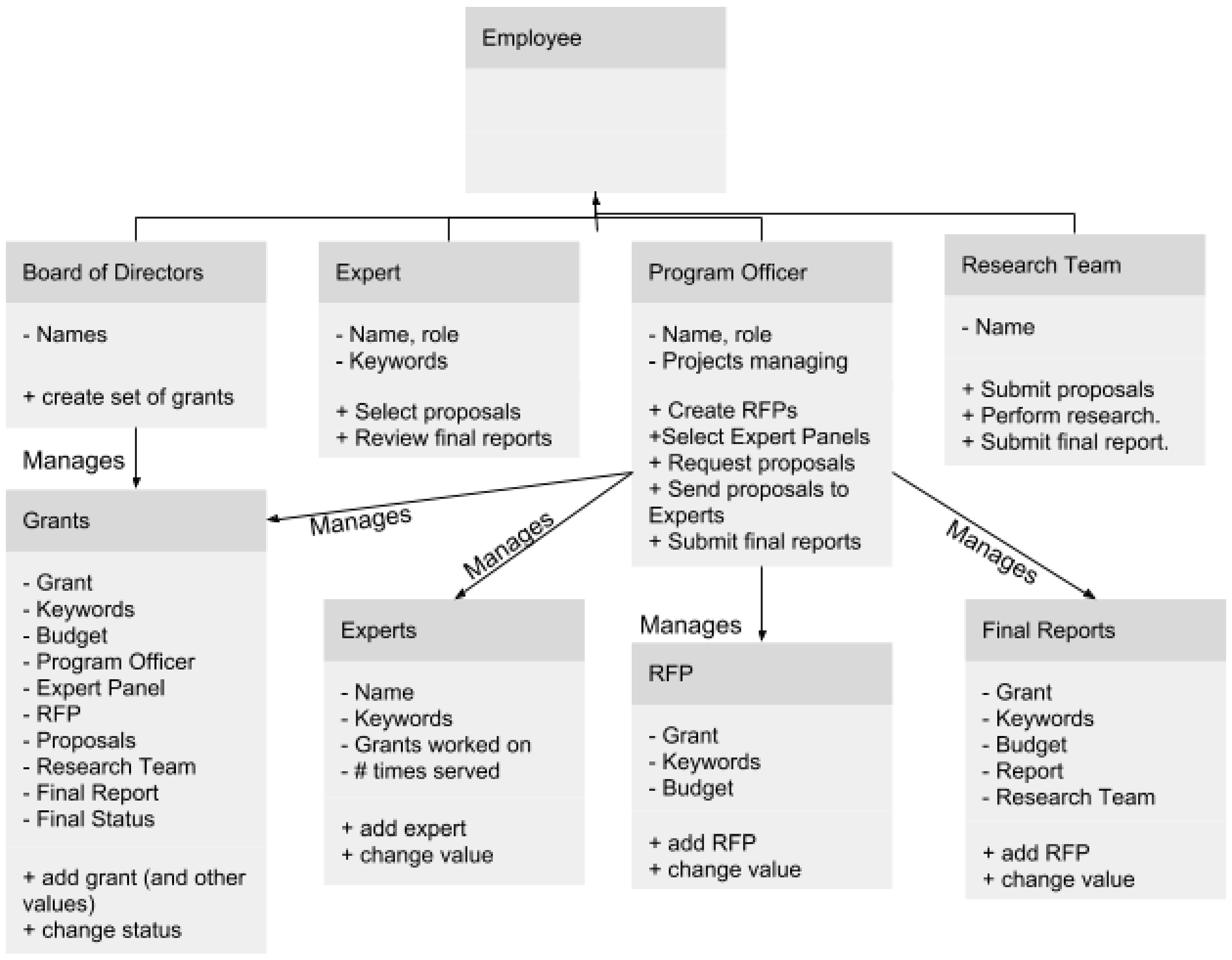
Relationships:

Generalization: Employee

Aggregation: projects (that they're working on)

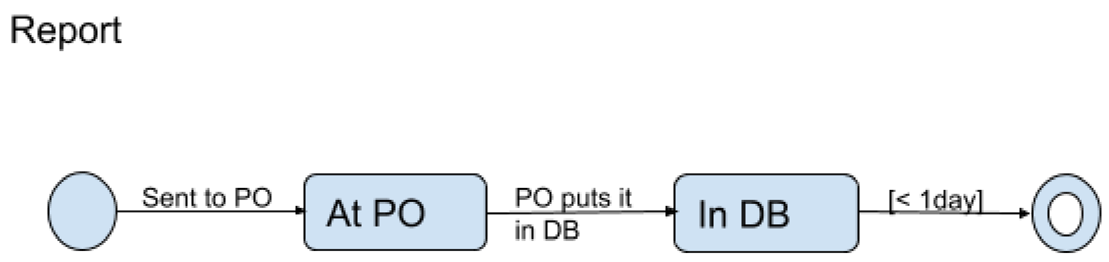
Other associations: none

6. Class diagram



We maintain classes for the people within the system, and the items that are managed.

7. Behavioral state machine



- ❖ **Behavior diagrams show:**
  - ➢ Changes that occur within an object.
  - ➢ The transitions that an object passes through the execution of a use case
- ❖ **Submit Report use case:**
  - ➢ An instance of the research team class, (aResearchTeam)
    - ■ Final Report sent to Program Officer.
    - ■ Final Report then entered in Final Reports DB

8. CRUDE matrix

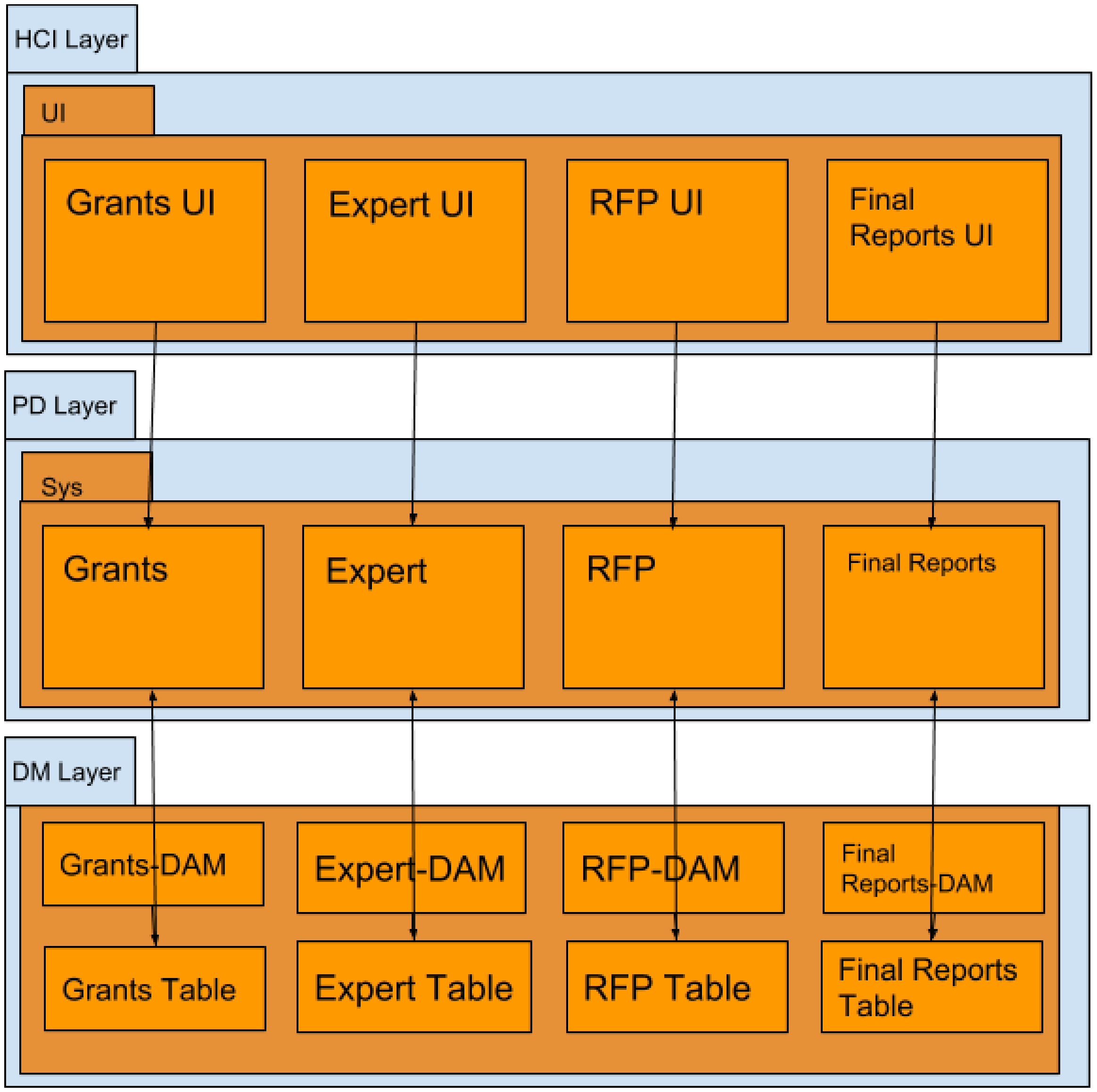
Crude Matrix

	Board of Directors	Expert	Program Officer	Research Team	Grants	Experts	Final Report	RFP
Board of Directors					C			
Expert					R		R,U	
Program Officer					R,U,D	C,R,U,D	C,R,U,D	C,R,U,D
Research Team					E		E	E
Grants								
Experts								

Final Reports								
RFP								

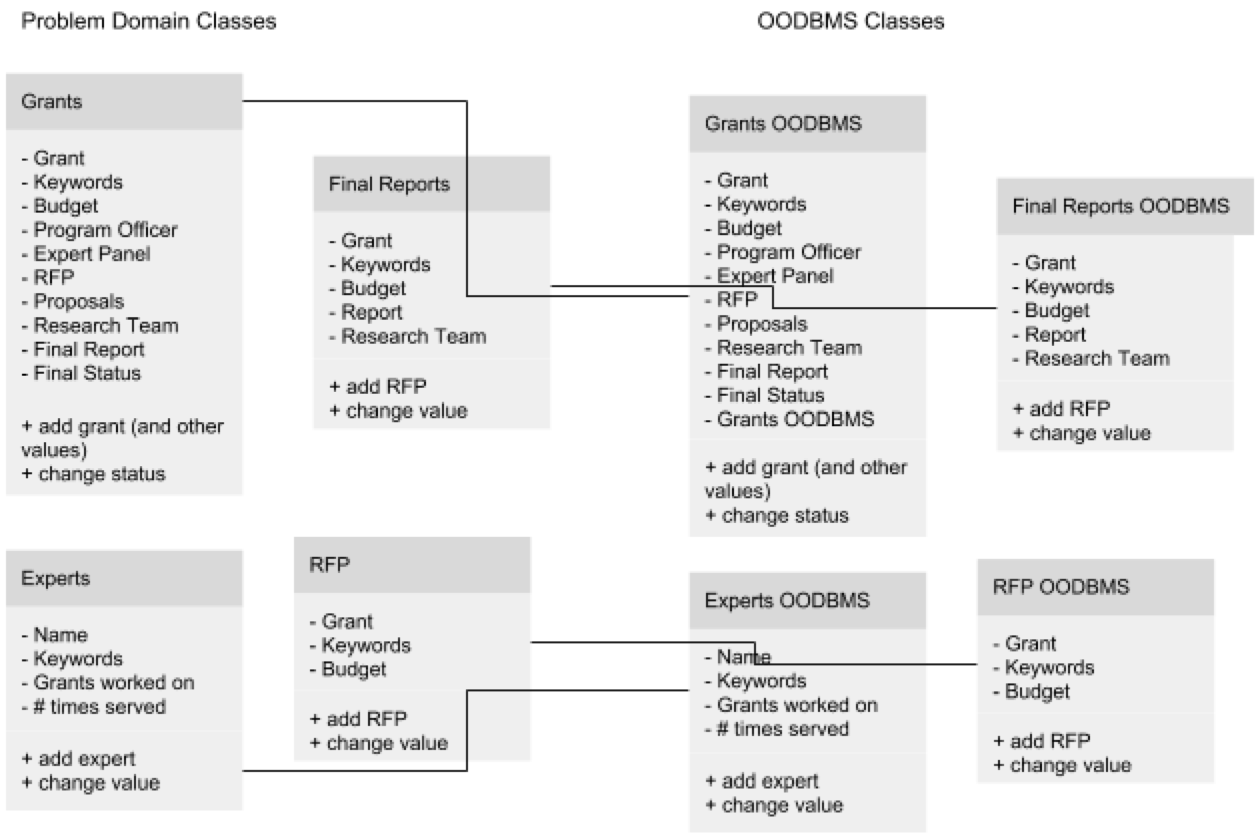
The Board of Directors is able to create grants, the experts are able to read the grants, the program officer is able to read, update, and delete grants, and the research team is able to execute grants. Program Officers are able to create new experts in the database, read data about the experts, update information about the experts, and delete information about experts. Experts are able to read and update final reports, program officers are able to create, update, read, and delete final reports, and research teams are able to execute final reports. Program Officers are able to create, update, read, and delete RFP’s, and research teams are able to execute RFP’s.

9. Package diagram



The packages represent the Human-Computer Interaction layer, the Problem Domain layer, and the Data Management Layer. The UI layer handles the presentation of the data, the problem domain layer handles the application logic of the data, and the data management layer stores the data and handles the the data access logic.

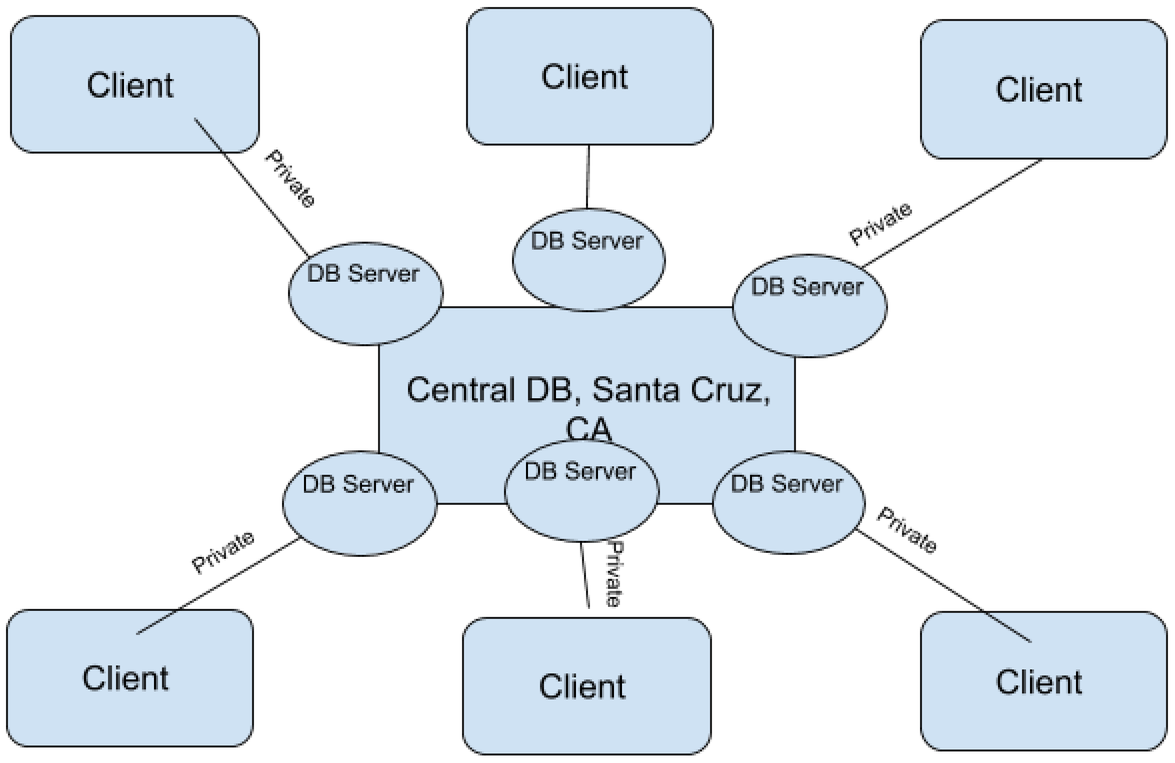
10. Mapping of problem domain objects to a relational database management system (RDBMS) format.



One database is being used to manage each problem domain classes data, so we need to maintain one dbms for each class. The dbms classes will have the same values and functions as the pd classes, so that the databases can be changed along with the classes themselves.

11. Deployment diagram.

The central db is held near company headquarters in Santa Cruz, and we will maintain database servers to access the data. We will not maintain any regional databases, as we will not experience great enough demand to require them. Clients will be thick, handling both presentation and application logic for the system. At a high level, the layout of the system is:



Because we aren’t using any regional servers, this diagram should suffice.