**CIS 375: Software Engineering I**

**Professor Steiner**

**Butterfly Tracking System**:

**-Part 1-**

Software Project

Management Plan

**By *Team* Straw Hats**:



**Hadi Nasser**

**Daniel Hamadeh**

**Venkata Suresh Machetti**

**Mohammed Alsanad**

**Zachary Menken**

-10/22/2015-

\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Table of Contents\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**1.0** Introduction...…………......………………………………….. Pg. **2**

* **1.1 Project scope**
* **1.2 Major software functions**
* **1.3 Performance/Behavior issues**
* **1.4 Management & technical constraints**

**2.0** Project Estimates……...……………………………………... Pg. **4**

* **2.1 Historical data used for estimates**
* **2.2 estimation techniques applied and results**
  + **2.2.1 Estimation technique *m***
  + **2.2.2 Estimate for technique *m***
* **2.3 Reconciled Estimate**
* **2.4 Project Resources**

**3.0** Risk Management……………………...…………………….... Pg. **7**

* **3.1 Project Risks**
* **3.2 Risk Table**
* **3.3 Overview of Risk Mitigation, Monitoring, Management**

**4.0** Project Schedule.......................................................................Pg. **11**

* **4.1 Project task set**
* **4.2 Functional decomposition**
* **4.3 Task network**
* **4.4 Timeline chart**

**5.0** Staff organization…………………………………………...Pg. **15**

* **5.1 team structure**
* **5.2 Management reporting & communication**

**6.0** Tracking & Control Mechanisms……………………......…Pg. **17**

* **6.1 Quality assurance & control**
* **6.2 Change management & control**

**7.0** Appendix………………………………………………………..Pg. **18**

1. **Introduction**

The task is to develop a butterfly tracking system. The program requires an interface that allows the user to enter their own personal data, as well as information on butterflies the user has input themselves or other information that has been put in by other users

* 1. **Project Scope**
* Data repository Display and an update of people/organizations reporting butterfly tag/sightings.
* Butterfly Tracking system Display and creation/update of a tagged butterfly.
* Butterfly tracking display and update for individual reports (no tag)
* Butterfly sighting history display base on dates and tag/no tag/both data types (graph)
* Given a date and/or Location, show a list of the tracking results
* Given a butterfly tag, list the migration route.
* Loading of daily tracking data based on butterfly tag
* Download of all/location/tag tracking data.

* 1. **Major Project Functions**
* Input

1. Data Repository Display
   * Reporter/Tagger Number – system assigned
   * Name
   * Address
   * Phone
2. Butterfly tracking display – tagged butterflies
   * Date of tagging
   * Location (city)
   * State/Province
   * Country
   * Latitude
   * Longitude
   * Who Tagged
3. Butterfly tracking display – (no tag)
   * Date of tagging
   * Location (city)
   * State/Province
   * Country
   * Latitude
   * Longitude
   * Who reported

* Processes

1. Database
2. All input is stored within a database
3. Database allows for instant access to the specified data that the user wishes to view/manipulate.
4. The database will safely store all data and all changes made to the data.
5. Program Functions
6. Error Checks
7. Program will check for invalid data entered.
8. Invalid data will not be stored
9. Queries to the database - Butterfly
10. Searches the database based on dates
11. Searches the database based on location (city/ states/ country/ latitude/ longitude).
12. Searches the database based on Tags
13. Queries to the database - People
14. Searches the database based on Name
15. Searches the database based on location info (address/Phone)
16. Searches the database based on Tags (reporter number/tagger number)

* Output

1. Invalid Data
   * all invalid data will be displayed as errors
   * invalid data will not be recorded in database
2. Valid Data
   * Requested information will be displayed to user
3. Errors
   * Error-Log: A log of all errors will be recorded and displayed to user if requested
   1. **Performance/Behavior issues**

Delay in program load/run is possible with a very large program.

**1.4 Management and technical constraints**

‘Drop Dead’ Dates:

* + Part 1 – due October 22,2015
  + Part 2 – Unknown
  + Part 3 – Unknown

If the project is not completed by the specified dates, the team’s overall grade will suffer.

Limited resources:

* Work time for the group

Technical constraints:

* + - Database compatibility issues between computers
    - Possible lack of database skills

1. **Project Estimates-not complete; see highlighted areas**

**2.1 Historical data used for estimates**

Class demo from a previous class’s project

* Gave a good grasp on the project’s scale
* Helped estimate total time for completion

Previous project experience from other courses

**2.2 Estimation techniques applied and results**

**2.2.1 Estimation technique m1:** Software development effort estimation

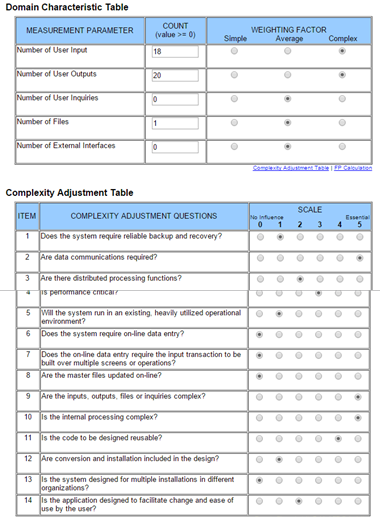
This is a raw analysis, predicting how many hours could revolve around our project efforts.

* + - Ideas / Brainstorming: 5-10 work hours per week
    - Documents / Documentation: 20 work hours per week on average (when applicable)
    - Building the database:
  + Structure/Diagram: 10 work hours
  + Creating Functions: 15 work hours
  + Testing / fixing: 30 work hours per week (when in testing phase)
  + Adjustments (sick days / project changes): 40 additional work hours

**2.2.2 Estimate for technique m1 effort estimation:**235hours.

**2.2.1 Estimation technique m2:** Function points

Function points (unit of measurement expressing the amount of business functionality a system provides to a user) measure software size.



**function points table**

**2.2.2 Estimate for technique m2 function points*:*** 243 hours.

**2.2.1 Estimation technique m3 Lines of Code:**

**-** Input: 700 LOC

* Processes: 700 LOC
* Output: 400 LOC

**2.2.2 Estimate for technique m3 LOC*:*** 1800 LOC. 1800 Lines of Code would take us around 300 hours.

**2.3 Reconciled Estimate**

Since the estimates seem close to each other, we average all 3 techniques’ estimates to present the estimation of the total Time and Effort cost of the program at 260 Hours.

Average estimated hours = (M1+M2+M3)/3 = (235+243+300)/3= 260 Hours.

**2.4 Project Resources**

**People**:

* + - Daniel Hamadeh
    - Hadi Nasser
    - Mohamed Alsanad
    - Zachary Menken
    - Venkata Suresh Machetti

**Hardware**:

* + - Desktops
    - Laptops / Macs

**Software**:

* + - Microsoft SQL – Database
    - Visual Studio – C++ coding
    - Google Docs – Document sharing
    - Microsoft Word – For Documentation
    - Microsoft Project – Diagrams / Charts
    - Microsoft Power-point – Presentations

1. **Risk Management**
   1. **Project Risks**

**Team Member Sick Days / Vacation Time**:

For any member who needs sick days / vacation time, their share of the work will be split up amongst the other team members during the team member’s time off. After returning, the team member who was gone will take on a little extra work to make up for his time away.

**Changes to the Project**:

Any changes made to the project will lead to extra work hours. To make up for this, the team will meet-up and discuss the required actions and adjust the workload accordingly.

**Conflicting Team member schedule**:

For any team member that cannot meet-up with the group due to schedule confliction, the team leader will update the team member on the project and assign the team member his required part for the week.

**Program doesn’t compile properly**:

Database is not as common in C++ as other software application languages such as C#. Since the team agreed on implementing a database, but are limited to the common language knowledge of C++, compilation issues may occur after much of the programming is completed for the butterfly tracking system. In this case, we’d have to either redo the program in C# (taking into account learning time), or implement file processing instead of the planned/designed database.

**Presentation of system failure**:

In the case that everything is considered working after testing is done, but suddenly during the presentation an issue occurs (such as database connection issue rises), we’d have a video demo available that could be displayed instead, showing the program working fine in all the different probable testing aspects.

* 1. **Risk Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Risk #  (from most (1) – lowest in vitality) | Risk | Probability risk will happen  (High, Medium, Low) | Impact  (Critical, High, Medium, low) | Mitigation  (impact reduction) Plan | Contingency  (emergency)  Plan |
| 1 | Changes to the project | High | High | To make up for extra work hours incurred, the team will meet-up and discuss the required actions and adjust the workload accordingly. | Even more work hours will be dedicated to adjust to the increased workload. |
| 2 | Conflicting team member schedule | High | Medium | The team leader will update the team member on the project and assign the team member his required part for the week. | If the team member becomes unresponsive, his/her workload will be distributed evenly between the other team members. |
| 3 | Program doesn’t compile properly (i.e. database issue) | low | Critical | Continue with increased amounts of testing to pinpoint the problem within the program. | 1. Redo the program in C# (taking into account learning time), or 2. Implement file processing instead of the planned/designed database. |
| 4 | Presentation of system failure | low | Critical | N/A | Have a video demo available that could be displayed instead, showing the program working fine in all the different probable testing aspects. |
| 5 | Team member sick days/vacation time | Medium | Medium | After returning, the team member who was gone will take on a little extra work to make up for his time away. | Split up share of work amongst the other team members during the team member’s time off. |

**3.3 Overview of Risk Mitigation, Monitoring, Management**

***Risk #1: Team member absence (sick/vacation time)***

Mitigation: Split up share of work amongst the other team members during the team member’s time off. After returning, the team member who was gone will take on a little extra work to make up for his time away.

Monitoring: Keeping close communication with each team member.

Management: Team member becomes too sick to stay focused; immediate attention of this risk will follow. Vacation time will be planned ahead of time, planning of the work load will be scheduled accordingly.

***Risk #2: Changes to the project***

Mitigation: To make up for extra work hours incurred, the team will meet-up and discuss the required actions and adjust the workload accordingly.

Monitoring: Immediate notification. Project changes are a direct result from the “Customer” (Professor) changing the project himself.

Management: Immediate attention of this risk is required, project changes are a top priority.

***Risk #3: Conflicting Team member Schedules***

Mitigation: The team leader will update the team member on the project and assign the team member his required part for the week.

Monitoring: Good communication between team members is required. Knowing each of the team member’s schedules every week will be important to the outcome of the project’s delivery date.

Management: Risk will be mitigated when the team member cannot make it to a weekly meeting. Action will be taken within 24 hours of the team member’s absence.

***Risk #4: Program doesn’t compile properly***

Mitigation: Continue with increased amounts of testing to pinpoint the problem within the program.

For an extreme case:

- Redo the program in C# (taking into account learning time), or

- Implement file processing instead of the planned/designed database.

Monitoring: Constant testing of the program will be run throughout all parts of the decomposed functions.

Management: Risk will be mitigated when the program becomes more and more unstable, where the program generally requires more testing to take place to fix it.

***Risk #5: Presentation of system failure***

Mitigation: Have a video demo available that could be displayed instead, showing the program working fine in all the different probable testing aspects.

Monitoring: After the program is completed and the testing phase is over, the team will practice the presentation on different hardware platforms to monitor the successful functionality of the program.

Management: Immediate action will be taken. Program will be presented on the team’s tested hardware to ensure a presentation.

1. **Project Schedule**

**4.1 Project task set**

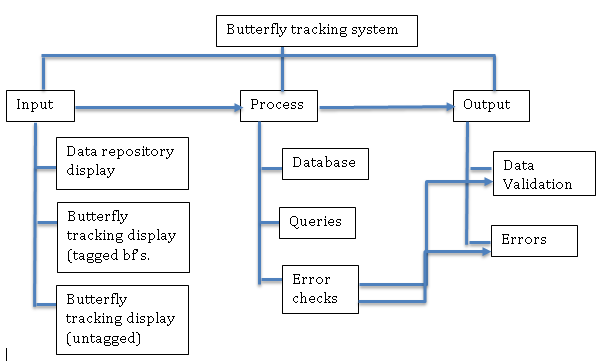
Task sets:

**Documentation**Software project management plan  
Decomposition diagram  
Entity relationship diagram  
Class Diagram  
Project change control document (Only when needed)  
Error log

**Coding Tasks**  
Design analysis (Program / Database)  
Building (Program/ Database)  
Testing (phase 1 / 2)  
Fixes / Repairs

**Presentations**  
Part 1 / 2 / 3  
Practice presentation

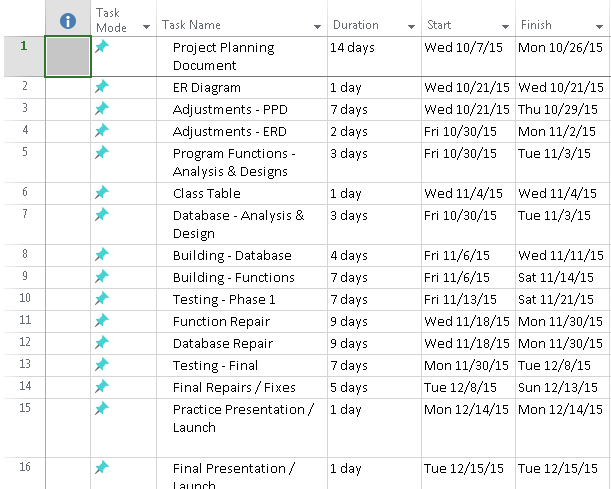
**4.2 Functional decomposition-This depends on tasks**

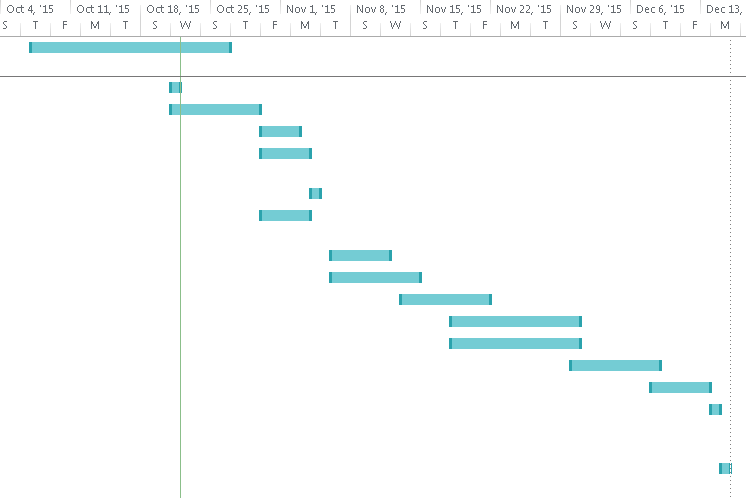


**4.3 Task network table**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Task | Task Description | Duration (Estimate in Days) | Predecessor Task | Successor Task | Delivery Date | Who is responsible | Who is working on it | % complete |
| A. | Developing the project planning document | 14 | N/a | B,C | 10/22/2015 | All team members | All team members | 100 |
| B. | ER diagram | 1 | N/a | C,D | 10/22/2015 | Mohammed & Zach | Mohammed & Zach | 100 |
| C. | Adjusting the planning document | 7 | A,B | D,E,G | 10/29/2015 | All team members | All team members | 0 |
| D. | Adjusting the ER Diagram | 2 | C | E,G | 10/30/2015 | Mohammed & Zach | Mohammed & Zach | 0 |
| E. | Program Function analysis & Design | 3 | C,D | F | 11/3/2015 | All Team Members | All Team Members | 0 |
| F. | Class Table | 1 | F | G,H | 11/4/2015 | Zach, Dan, Suresh | Zach, Dan, Suresh | 0 |
| G. | Database analysis & design | 3 | C,D,G | H | 11/7/2015 | Zach, Hadi, Mohammed | Zach, Hadi, Mohammed | 0 |
| H. | Build the Database | 4 | F,G | J | 11/11/2015 | Zach, Hadi, Mohammed | Zach, Hadi, Mohammed | 0 |
| I. | Build the Functions | 7 | E,F,G | J | 11/14/2015 | Dan, Suresh | Dan, Suresh | 0 |
| J. | Testing Phase #1 | 7 | H, I | K | 11/21/2015 | All team members | All team members | 0 |
| K. | Function repair | 9 | J | M | 11/30/2015 | Zach, Dan, Suresh | Zach, Dan, Suresh | 0 |
| L. | Database Repair | 9 | J | M | 11/30/2015 | Hadi, Mohammed, Zach | Hadi, Mohammed, Zach | 0 |
| M. | Final Testing Phase | 7 | K,L | N | 12/8/2015 | All team members | All team members | 0 |
| N. | Final Repairs / Minor fixes | 5 | M | O | 12/13/2015 | All team members | All team members | 0 |
| O. | Group Presentation Practice | 1 | N | P | 12/14/2015 | All team members | All team members | 0 |
| P. | Launch Program / Presentation | 1 | O | n/a | 12/15/2015 | All team members | All team members | 0 |

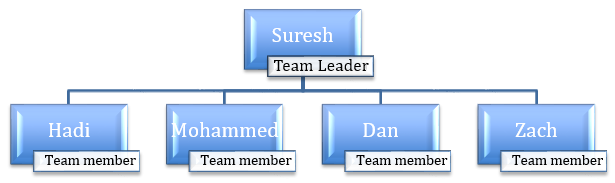
**4.4 Timeline chart**





1. **Staff Organization: assign more roles based on tasks above**

**5.1-Team *Straw Hats* Structure**



**Team members’ Roles**

**Suresh:** As a team leader, Suresh will deliver a weekly progress report of where we are on the project planning/program. He will also overlook the whole team and make sure work is getting done accordingly and on time. He will also be involved in the program development in C++.

**Hadi:** Contributing to the planning and development of the system, as well as forming the database.

**Mohammed:** Contributing to the planning and development of the system, as well as forming the database.

**Dan:** C++ programmer/ Lead documenter. Mainly in charge of creating the needed documents for the group, as well as working as a backup programmer.

**Zach:** C++programmer/database developer. Zach will work on both the database and the main program to ensure top quality between both areas.

**5.2 Management Reporting & Communication**

**Progress Reporting**

Suresh, the team leader, is responsible to turn in a weekly status/progress report of where we are on the project.

**Team Inter/Intra Communication**

As far as managing our inter-team communication, we share a common understanding of what is going on, what needs to be done, responsibilities assigned to each person, and comprehension of the documentation and diagrams.

Although we may not be meeting each other regularly (an intra-team communication aspect), we are able to communicate often through group messaging, which is convenient in our case where we aren’t as much available to discuss the project in the same location. This way, we are able to monitor our progress regularly via message notifications, and we can focus on more detailed things in the limited time we are able to meet each other weekly.

Google Docs is another way we manage to contribute together to the project documentation and planning, whether it be concurrently or at our time’s conveniences; this way, we are all able to keep track of where we are, and the team members can edit the same documents towards completion.

**6.0 Tracking and Control Mechanisms**

**6.1 Quality assurance and control**

Traceability will play a big part in assuring the program’s full success. During testing, if errors come up, we’ll be tracing back and forth between our planned functionality, and fix any errors in doing that.

We’d also keep track of the errors by an error log, in which errors are recorded, as well as the solution to each; this way, if similar errors appear later on, then solutions documented for previous similar errors will be utilized to save time.

**6.2 Change management and control**

All Change control and management will be delegated through the Team leader. In the event that a change is requested, it will be the team leader’s job to decide whether the change is major or minor and afterwards the team itself will follow the documented plan.

**Minor Change Control:**

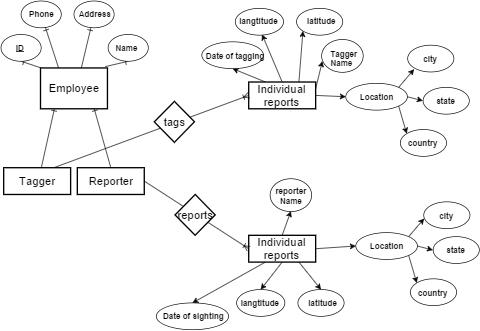
For minor changes the team will comply as long as the amount of minor changes remains reasonable. If too many minor changes are requested, the team will group them as a whole, and treat it as a major change.

**Major Change Control:**

For Major changes, the team will have to meet and discuss the impact that the changes will have to our project. The team lead documenter will then create a change control document that lists problems the major change will bring, and present it to the person(s) who requested said change. This will make sure that person is fully aware of the software’s current quality level and possible downgrade in quality / increased delivery date the change could bring. Afterwards if the change is still desired the team will ultimately comply and continue to work on the software.

**7.0 Appendix**

**Butterfly tracking system Complete ERD**



**Documentation supplements**:

Function points calculator: **TINY TOOLS**, URL: <http://groups.engin.umd.umich.edu/CIS/course.des/cis525/js/f00/harvey/FP_Calc.html>

Project plan template: <http://www.rspa.com/docs/projectplan.html>