Online Health Monitoring System

Project Legacy

COP 4331, Fall, 2014

Modification history:

Version	Date	Who	Comment
v0.0	8/15/00	G. H. Walton	Template
v1.0	9/1/2014	C. N. McCue	First Revision

Team Name: Team 14

Team Members:

- Jon Carelli - email - web page
- Chris McCue - email - web page
- Chris Chaffin - email - web page
- Ethan Pitts - email - web page
- David Gundler - email - web page
- James Luke - email - web page

Roles

Since team formation and project selection took two weeks, it is considered Phase 0. Phase I consists of Deliverables I. Below is the work breakdown of Phase 1.

Deliverables I Work Distribution	**Team member(s)**	**Work Percentage**
Concepts of Operations	Ethan, Chris M.	50%, 50%
Software Requirements Specification	Jon, David	50%, 50%
Project Management Plan	Chris M.	100%
Test Plan	Chris C., James	50%, 50%
Video Chat Research	James	100%
Quality Assurance, Revision and Compilation	Chris M.	100%

Phase 2 is synonymous with Deliverables II. Below is the work breakdown of Phase II.

Deliverables II Work Distribution	**Team member(s)**	**Work Percentage**
High-Level Design	Chris M.	100%
Detailed: Database Manager/Login/Settings	James	100%
Detailed: Alert Manager	David	100%
Detailed: Navigator	Ethan	100%
Detailed: Chat Manager	Jon	100%
Detailed: Pill/Appointment Manager	Chris M.	100%
Detailed: Calendar	Chris C.	100%
Quality Assurance, Revision and Compilation of DII	Chris M.	100%
Presentation Slides	All	100%
Presentation Outline, Revision and Compilation	Chris M.	100%

Although source code is part of Deliverables III, we consider coding as a separate phase. Phase 4 consists of coding. Below is the work breakdown.

Code Work Distribution	**Team member(s)**	**Work Percentage**
Research	All	100%
Database Manager/Login/Settings	James	100%
Alert Manager	David	100%
Navigator	Ethan	100%
Chat Manager	Jon	100%

Pill/Appointment Manager	Chris M.	100%
Calendar	Chris C., Jon	50%, 50%
Database Integration Help	James	100%
Code Integration	All	100%
Unit Testing	All	100%
Integration Testing	All	100%
System Testing	All	100%
Refactoring	All	100%

Phase 3 consists of Deliverables III. For source code, look at the above table. Below is the work breakdown for Deliverables III.

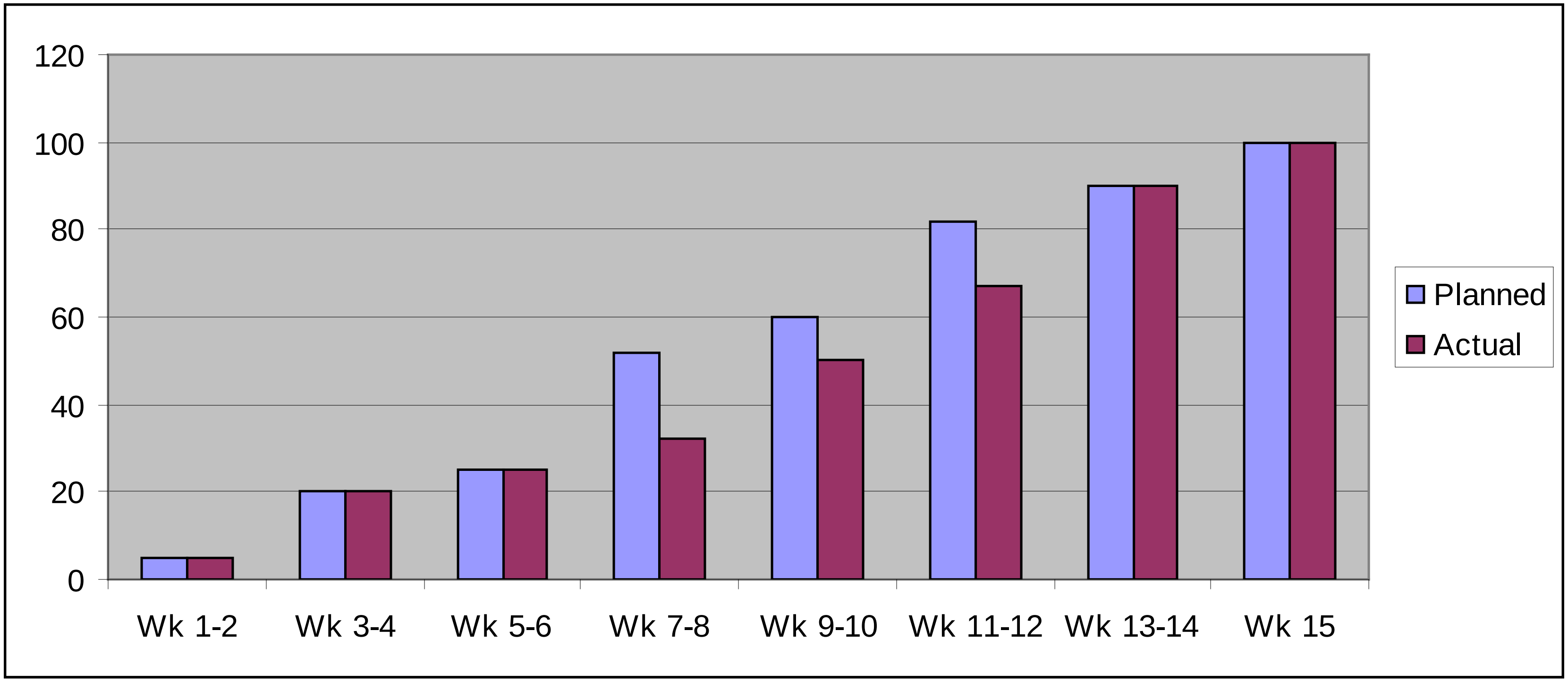
Deliverables III Work Distribution	**Team member(s)**	**Work Percentage**
"User Manual" per each module	All	100%
User Manual Revision and Compilation	Ethan	100%
Build Instructions	James	100%
Project Legacy	Chris M.	100%
Test Results per each module	All	100%
Test Results Revision and Compilation	Chris C.	100%
Quality Assurance, Revision and Compilation of DIII	Chris M.	100%
Presentation Modules	All	100%
Presentation Revision and Compilation	Chris M.	100%

Below are all the routine administrative tasks.

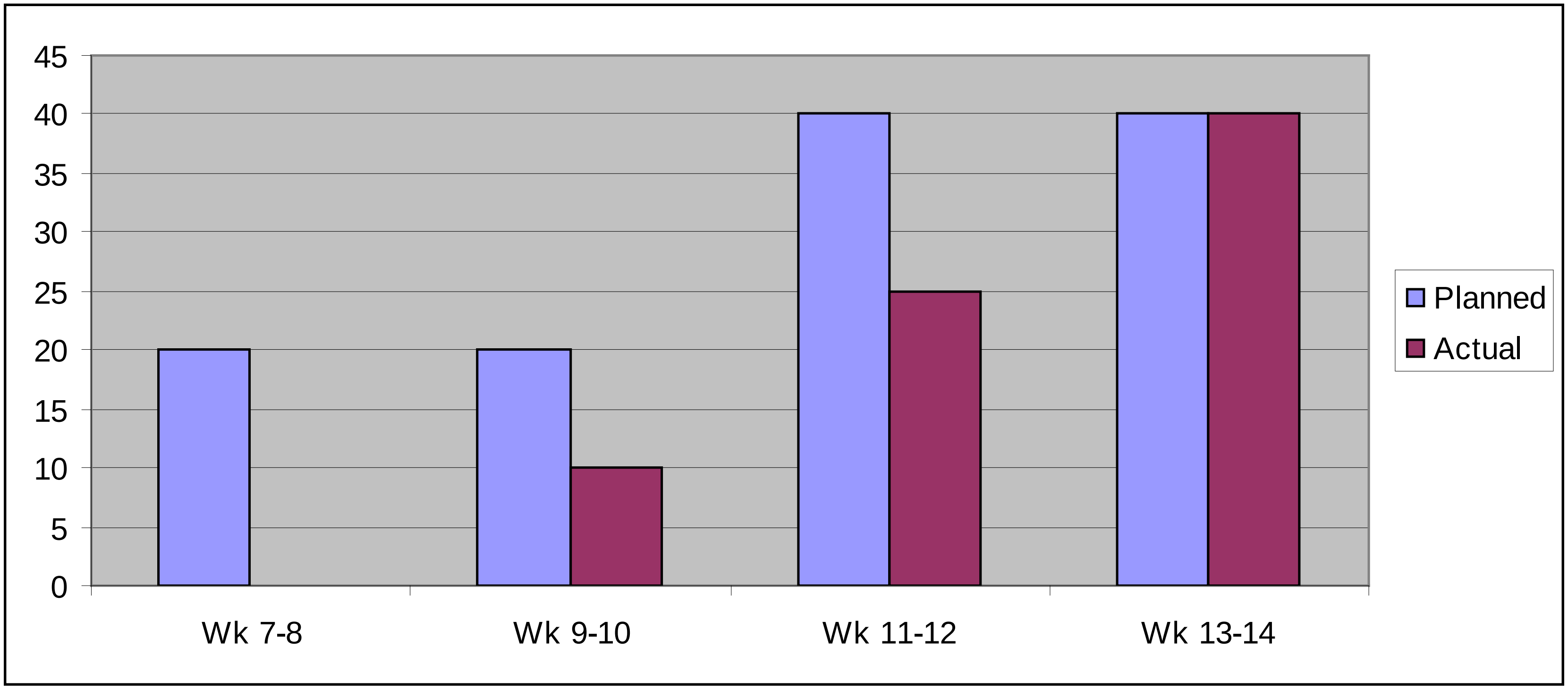
Routine Task Work Distribution	**Team member(s)**	**Work Percentage**
Analysis, Bi-weekly Project Reports and Log Tally	Chris M.	100%
Meeting Agenda and Minutes	Chris M.	100%
Website Administration	Jon, Chris M.	60%, 40%
Individual Websites	All	100%
Liaison	Chris C., David	100%, 100%
Quality Assurance	Chris M.	100%
Configuration Management	All	100%

Analysis

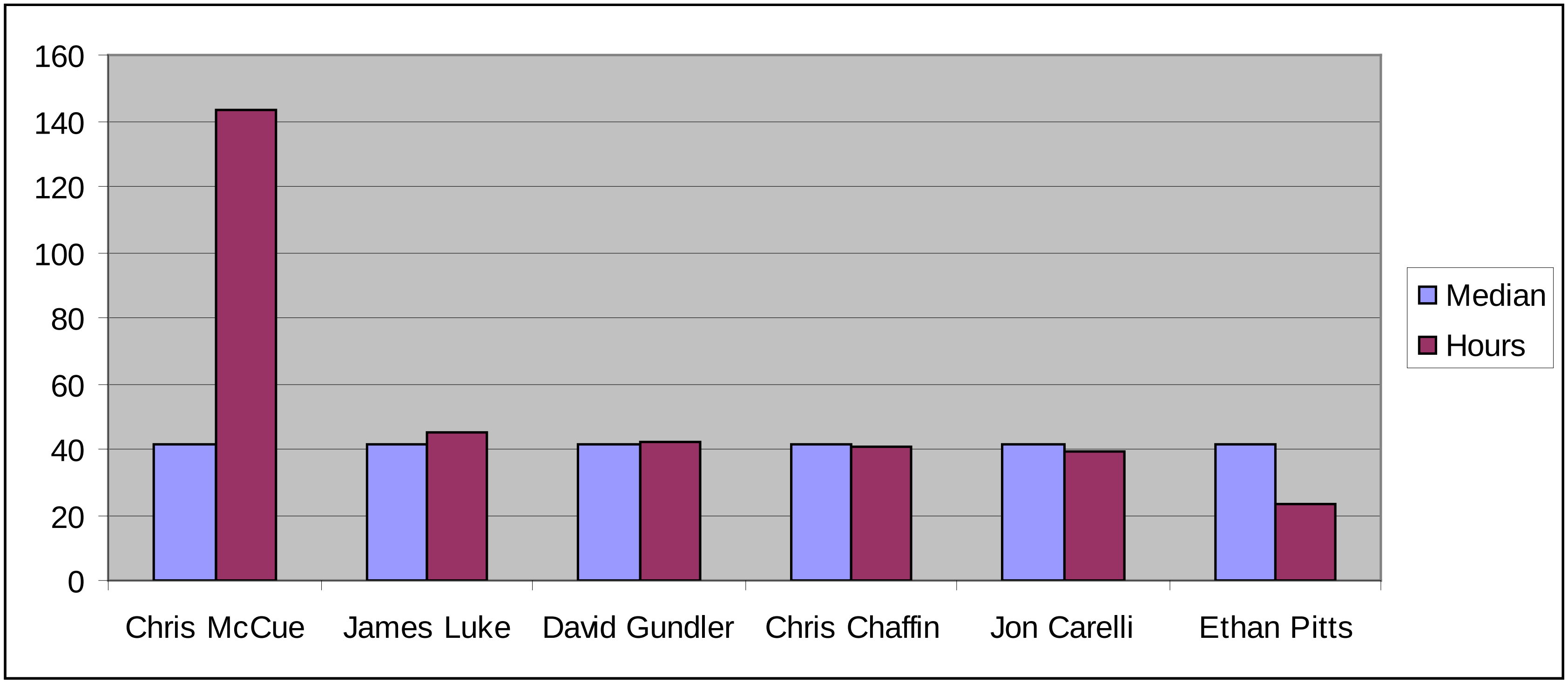
The chart below compares our overall progress metrics from the bi-weekly reports with the expected overall metrics. As you can see, the progress was fairly steady. Except for weeks 7-12, we were on schedule throughout the project. Our progress was behind for weeks 7-12 due to coding. During those weeks, our documentation was on schedule.



The chart below takes a closer look at coding for weeks 7-14. This was the only major issue we encountered. Though we were behind, we rapidly caught up by week 14.



The chart below shows the hours worked per team member. Most team members were close to the median of 41.25 hours. Chris McCue was extremely overworked, while Ethan was underworked.



- Assessment of the Quality of the Final Product:

  The Online Health Monitoring System functions according to the “must have” requirements. The pill/appointment manager adds pill schedules and appointments for patients. It is robust. Pill schedule error checking consists of: no pill name entered, invalid dates, end time is less than start time and duplicate pill names. Appointment error checking consists of: invalid dates and duplicate appointments. The Calendar is updated with pill schedules and appointments. The Alert Manager sends pill schedules and appointment SMS alerts. The Chat Manager allows for doctor-patient and patient-doctor communication. The project works well. Everything is as expected. One can login to the website specified on the User’s Manual and use all the functionality.

- Recommended Use of the Final Product:

The project meets the specified requirements. However, it would need modification for use in the real world. Currently, a patient can have one doctor, according to our requirements. More security would be required to make sure no one fakes being a doctor. These are examples of minor changes that would need to be implemented for real world use. The risk of lawsuits with medical applications is so great that we would hesitate to use this in the real world. However, in the university world, this is delivered as requested.

- Known Problems:

All work was finished to meet project requirements. There are only two minor known problems. The calendar of newly registered users shows the most recently logged in patient's pill and appointment schedule until the new user gets adds a pill or appointment scheduled. Potential fixes include: New users automatically getting a dummy pill assigned to them once they register. Change algorithm to only display pills/appointments if a value is returned from database. If not, display nothing. Also, the SMS spam filter is sensitive. When the SMS code is turned off, that day's information will be lost.

- Adherence to Project Plan:

Overall, the team was able to adhere to the project plan very well. Administrative and technical tasks were divided equally among team members. Communication was maintained weekly with individual logs, emails and meetings before Deliverables. Corrective actions were taken as needed. For example, corrective actions were taken with the Test Plan. During revisions of deliverables, there was quality assurance to make sure standards were met on documentation. Risks were minimized by managing risks early on.

Overall, the team followed project time estimates well. Weekly logs included progress metrics. This made it easy to assess weekly progress. Occasionally, factors, such as homework in the Processes class or tests in other classes, may have delayed items a day or two. However, a cushion was built into time estimates to easily allow for this. Therefore, we were able to accommodate these minor deviations. Much of the time was spent just keeping up with the high priority items, such as next deliverable, presentation or meeting deadline. Therefore, coding had to be delayed a few weeks because of Deliverables II. It was suppose to be in parallel with Deliverables II. This was our only major deviation.

Because of this, we were not able to adhere to our planned software life cycle process well. We were not able to spend much time in prototyping stage of incremental phased development. We had to go straight for integration. Our process more closely resembled agile development.

The root cause of deviations from time estimates was the daunting task of documentation. The only major time deviation was coding. This was delayed because of the effort needed for Deliverables II. Since source code was not due until Deliverables III, it was lower priority. Besides this, there were only minor deviations of a day or two because of work load outside the project. Lower priority items did have a time lag. Outside of analysis of individual logs to assess progress and take corrective action at crucial points of the project, the Project Manager found the weekly analysis and bi-weekly project report to be a cumbersome, time-consuming activity of little worth. The Project Manager had a good sense of project time management without these reports and found them superfluous. Therefore, these low priority items often had to wait until all higher priority items were completed.

This delayed reports for weeks. Configuration management also lagged. It took a while to post documents on the website after their completion. This was considered a lower priority item to be done when time permitted.

- Defect Analysis:

There were a total of 44 defects. One defect was introduced, discovered and corrected during the code prototyping/unit testing phase. This was an email error. There were 38 defects that were introduced, discovered and corrected during the code integration/integration testing phase. Most of these errors occurred because of Database or Navigator integration. Until our code was integrated with the database, we could not really test our modules. That is why so many errors were discovered in the integration testing phase. There were 5 defects that were introduced during the code integration stage, but discovered by other team members and corrected during the system testing stage. These errors consisted of David finding errors with the Pill/Appointment Manager and Chris finding errors with the Calendar. In summary, most of our faults were caught and corrected in the same phase of the project. Also, most of our faults were due to integration. This was the most challenging stage of the project.

- Quality Assurance:

In the end, quality assurance was performed by the Project Manager during deliverable revision. This was sufficient to produce documentation in line with team quality standards. This was done in a timely manner before submission of deliverables. However, it was a burden for the Project Manager. This could have been improved if there were a person dedicated to quality assurance. However, this person would have to have a good sense of documentation. This is a rare quality to find among computer scientists and engineers.

The testing activities for our code were sufficient. Testing activities were performed in parallel with coding. Each module coder tested his code as he wrote it. Integration revolved around the Database Manager and Navigator. Integration testing was done by the module coder as we first integrated with the Database Manager and, then, with the Navigator. The tests were as timely as they could be, given that we were rushing to finish code before the demo. System testing occurred organically as we were putting the finishing touches on our modules. Given the time constraints of this project, our testing was optimal. If we had more time for this project, we would have each team member independent inspect the whole system for faults.

- Configuration Management:

Every team member was responsible for configuration management of his documentation or code. The team website was used as a hub for all documents. The author retained a copy as well. Overall, these were adequate for our small project in the short project life cycle. However, there was always a lag in the posting of documents on the website since this was considered a lower priority. There was only a minor problem with configuration management of the SRS document when three of us were modifying it at the same time. We were able to revise the SRS properly. For a larger project, our configuration management would need improvement. We could have improved configuration management by using stricter guidelines, such as a document change form, and/or having a dedicated person for configuration management.

For source code, we uploaded our source code to the server during development. For final source code, we used Dropbox as a repository. We never experienced any configuration management problems with our code.

- Suggestions for the Future:

Given the current team dynamics, the author feels that the technical processes put into practice were optimal. Time management and communications were handled well. However, these practices did cause overload for the author. This would be an area for improvement. No advice can offered as to how this would be implemented as the author has not yet found a way to do so.

For future project teams, advice would be as follows:
Start early with team formation and project selection.
Set modest and realistic expectations for your project.
Set deadlines with the expectation of procrastination being the norm. In other words, make early deadlines before real deadlines.
Make quality assurance and team expectations clear at the beginning.
Meet and communicate frequently (at least weekly through individual log).
Follow-up on progress through progress metrics in individual logs and remind frequently of upcoming action items.

If assigned a project 10 times the scope of this project, work distribution would need to be more specific and guidelines would need to be more specific. Tasks, such as configuration management, quality assurance and web maintenance, would need to be assigned to specific individuals. There would need to be meetings just to discuss guidelines and expectations, such as those in the Project Management Plan. Much more time would be needed to carefully articulate Deliverables I. The software life cycle process would be followed more strictly. In summary, more time would be needed before the design phase.

For a project 100 times the scope of this one, even the minutest tasks would require dedicated personnel. For example, there would be an individual dedicated to the project website. The team structure would need to be more hierarchical. There would be teams for design, teams for coding and teams for administrative tasks. A programmer may only have a vague idea of the big picture as tasks are subdivided again and again. For example, a programmer may work on a few classes of a module. The coder would never work on anything from Deliverables I. Deliverables III would become much more important as the code would need to be maintained. Code maintenance would be an important item to consider.
