Final Comprehensive Systems Assignment

CEGEP HERITAGE | CHRISTOPHER FOX

Philip Dumaresq

Systems III

2017

Contents

[Introduction 2](#_Toc500175579)

[Estimation 2](#_Toc500175580)

[Estimation Methods 2](#_Toc500175581)

[Chosen method 4](#_Toc500175582)

[Risk Management 4](#_Toc500175583)

[Velocity 5](#_Toc500175584)

[What is velocity 5](#_Toc500175585)

[sprint 1 6](#_Toc500175586)

[velocity comparison 6](#_Toc500175587)

[Standards 6](#_Toc500175588)

[Why are they important? 6](#_Toc500175589)

[Our Standards 7](#_Toc500175590)

[Security 7](#_Toc500175591)

[Authentication and Authorization 7](#_Toc500175592)

[Encryption 8](#_Toc500175593)

[Confidentiality and Privacy 8](#_Toc500175594)

[Database security 8](#_Toc500175595)

[Directory manipulation 8](#_Toc500175596)

[Injection attacks 9](#_Toc500175597)

[Defects 9](#_Toc500175598)

[Exploratory Testing 10](#_Toc500175599)

[Black Box Testing 10](#_Toc500175600)

[System Testing 11](#_Toc500175601)

[White Box Testing 12](#_Toc500175602)

[Static 12](#_Toc500175603)

[dynamic 13](#_Toc500175604)

[code coverage 13](#_Toc500175605)

[Installation Conversion Strategy 13](#_Toc500175606)

[Conclusion 13](#_Toc500175607)

# Introduction

The purpose of this report is to discuss the development of the Alumni system for Heritage College. The report will cover different types of project duration estimation techniques, and recommend which technique should be used by this system. Following that we will go into the velocity of our development team and how we will figure out what our velocity will be as a team. After that, we’ll get in to discussion of writing actual code by deciding on standards that our code will follow. These means code conventions that will be followed as well as styles that will be implemented in the user interface design. Included in this portion will be a short paragraph on why coding conventions are important, why should use them and why they *will* be followed by this team. Following that will be getting into testing the software, starting off with security. It will cover the primary security concerns with this system and how the system should be designed to handle the important issues. Still in testing, there is a section on defects and how you will write a proper bug report so that the testing can be effectively done. However, before writing a bug report, we need to know where to test and what types of testing will be done. So once proper bug reports have been discussed, different testing methods that this system will implement will be covered, including black box and white box testing techniques, system testing, and the possibility of doing some exploratory testing. Finally, the report will conclude with a section on the installation conversion strategy that we will implement and why this is important for the Alumni system.

# Estimation

## Estimation Methods

Function point estimation is when you take a total all of the required inputs, outputs, queries, files, and program interfaces and try to give each of them a difficulty, where that difficulty has a multiplier on it for time. So you multiply the number of low complexity inputs by the time multiplier for low complexity inputs and add that to the total time. The sum of all of these times is going to end up being the total unadjusted function point. You then adjust these points according to what that task is trying to accomplish, and add everything together again to get the total number of function points. For each programming language, you can then figure out the average number of lines of code for a single function point, so for example Python has ~21 lines of code on average for a single function point. So, if you have 15 function points, you’d multiply that by the number of LOC per function point (15\*59 = 885), and you’d get an estimate of how much code needs to be written for everything. This method of estimation is less accurate and will usually just be done by one person sitting down at a computer entering inputs for the numbers on everything and trying to determine how much everyone will have to do.

Function point estimation is highly unreliable as a method to estimate since there’s far more things to factor in than just some estimated lines of code to write to design a system. Use case points estimation is done to replace the function point estimates. Use case points is done by first designing all of your use case diagrams for your system and assigning complexities to things. For example, each of your actors will be assigned to be a simple, average or complex actor. A simple actor would be something along the lines of another system that can be interfaced with very easily by a well-defined API, and a complex actor would be a person that interacts with a UI. Each use case is the categorized by complexity as well. However, use case estimation also factors in external things that will affect the system. I will factor in the difficulty of the programming language, the level of motivation that the team has, the amount of part time staff working on the system, the familiarity with technology that your team members have, and other external things that may slow you down. It will then take all of these complexities of actors, use cases, and severity of external issues and it will calculate the amount of man-hours required to complete the system. While this pseudo-science method of determining the time required is far more accurate than just checking lines of code, these predictions are still very hard to accurately make. This is where the team will come into play to determine the time instead of just crunching some number arbitrarily.

Poker planning is an effective planning strategy where you get everyone together on a team and everyone is assigned playing cards with a series of number on them, often following some sort of Fibonacci sequence of number. I.e. 0, 1, 2, 3, 5, 8, 13, 21…100,?, where 0 means that the task will take a negligible amount of time to complete, and ? is a task so large that estimating its size isn’t possible. Tasks like this should usually be broken down into smaller sub tasks since the task is far too large in its current state. The team will go through each task 1 by 1 and everyone will assign a card to that task. The upper and lower bounds of estimations will justify why they think the task is so far to one side, and the team will take a vote on how long that task will take, and then it will get assigned to that length. Poker planning is quick and effective for getting a census among the team for your task duration.

Table top planning is similar to poker planning, with one major difference. Before going and assigning an actual value to each task, all the tasks are first pulled out of a pool and sized relative to each other. This is to help put the whole system into perspective as to what will take the most or least time to complete. This makes the process slightly more time consuming, but yields more accurate results than table top planning.

## Chosen method

The method that our team will be using for this system will be the table top planning method. This method is chosen for multiple reasons. First of all, using a method where the entire team participates means that we’ll have more accurate results than just having one person go off inputting number into a formula. Second of all, with this being a new team, we can use this as a way to figure out how the development team works together, and how well they communicate and get their points across effectively.

# Risk Management

This project, as with any, has come to us with some risks associated with it. The first major risk with this project is that we are going to be developing this project in Python, which is the requested technology to be used. In our team, we only have 2 developers who are familiar with Python, so for most of the team this is going to be a large obstacle. A second risk that this project faces is that we have one developer who will only be working 50% of the time, and another who needs to take off the entire month of February. This means that during this time, we are only going to have 3 full time developers on this project, and most of the time only 4 members will be full time. From the initial 7 full time developers that this team was hoping to staff, we are at a serious deficit. The final major risk in this project is that we are a new team with young developers. None of us have worked together before, and therefore we have no idea what the team dynamic is going to be like. We don’t know anything about how we work together as a team, and what kind of clashes we’re going to have as a team. We also have no idea what this team can accomplish in a single sprint. The team is comprised of relatively in-experienced developers as well, meaning we can’t make estimations based off previous experience. Now, we do have 6 weeks of slack time in this project, so there’s a lot that we can do to manage our risks effectively.

Each of the above poses a large risk on its own, but there’s a lot that we can do in hopes to mitigate the consequences of these risks. Python is a relatively simple language, and as such will not be too complicated for the team to pick up. In the first sprint in the project, we will dedicate more time to programming tasks assigned to those who are unfamiliar with the language and less total tasks to those who are. That way, those who know Python can act as a learning resource for those who are unfamiliar with it. For trying to find a little bit more about our team dynamic, there’s a lot that we can do. The main point I’d like to touch on is that we have our daily scrum meetings where everyone shares their current state. This can be very beneficial to the team because it makes us get together every day and helps us communicate face-to-face, where communicating over just email can be considerably more difficult for some. Having an open environment to communicate our concerns or needs will help the team grow together much more closely. Finally, to address the fact that we have employees who will not be following the regular work schedule, we have 6 weeks of slack time in the project. This means that we can allocate our resources such that all of our full-time employees are allocated 100% of their time during the full duration of the project, including those extra 6 weeks. They can be assigned more work to make up for it. The part time developer can have their task sizes increased due to the fact that they won’t be worked on full time, and other developers can take over when vacation time is taken. Overall, while each of these do present risks, taking these steps to mitigate them should be effective.

# Velocity

## What is velocity

Velocity is a method used to determine how quickly your team is able to complete tasks. Teams that have a higher velocity are on average able to complete more function points in a system in a single sprint than teams with a lower velocity. This is of course very subjective since the number of function points for a single sprint can vary depending on the team that did the initial estimation and created the function points for the sprint. So because of this, velocity is something that can’t really be compared from one team to another, it’s something that can really only be compared among one team from one sprint to the next or from one project to another. Each team’s velocity is unique and if you aren’t a member of that team can mean very little.

## sprint 1

Since this is a new team and we don’t know what our velocity will be, we’re going to underestimate how much we can get done in sprint 1. This way we can get an undershoot of what our team’s initial velocity will be, and as our sprints keep going on, we will continue to add work to the sprint backlog until we find our team’s true velocity. Once the amount of work we can get done in one sprint plateau’s out, we know that we will have our team’s work capacity and velocity.

## velocity comparison

As mentioned, a team’s velocity is very tightly coupled with that team itself. It’s something that can’t really be compared with other teams. So another manager suggested that for our initial sprint planning, I use their team’s velocity as a waypoint for determining our own. This is something that can’t be done, because his team can be very good and therefore assign very few points to something that realistically would take our team longer, or they could underestimate large tasks. These are things that contribute to a team’s velocity, and so it’s something that can’t be inherited from another team. What we’re going to do to plan our first sprint is we will all together as a team, think about what we think we can do in a pessimistic situation, and we’ll use that as our baseline and see how we do. We will use that as our initial velocity, and not that of the other dev teams’.

# Standards

## Why are they important?

Standards are a very important part of software development. Coding standard make code more readable for all developers, and not just your personal preference. They make the code look like it was written by a single developer in one sitting, meaning that everything is uniform and there’s no surprises when opening something new. Any IDE will have also have formatter settings, so you can code it in the way you want and just make sure it abides by the standard before checking your changes.

## Our Standards

One of the most important things to standardize in code will be the class names. Having a set naming convention for class names will make the code significantly more understandable as all names will give the user an understanding of what the method does. To avoid confusion with the database layers and the code, the database table names will follow the same naming the classes and each data table will have the same lists of properties/attributes with the same names. As the system is being built for Heritage College, it will also use the .NET framework as per the school’s standards, meaning all code will be written following the Microsoft standards. There will also need to be directory structure standardizing so that we keep our solution clean and organized. Every file will be placed in the proper directory to avoid name spacing issues and maintain organization.

In terms of standards for the user interface, there’s several that we need to follow according to the college standard. First of all, the college has a look and feel that they’ve got implemented across all of their websites and systems. We need to abide by this look and feel, implementing the proper color scheme and layout for our system. We also need to follow proper naming conventions for pages. We should have all pages following a simple convention to let the user know what page they’re on without the page name being confusing. Another standard that our user interface should follow is layout for input fields. For example, we don’t want labels for a text box being right aligned on some pages, left aligned on others and on top of the boxes on others. Unless it makes far more sense to break that convention for readability or mobile access, all forms should follow a standard. While mentioning mobile, there should also be separate standards in place for the system’s mobile design and for the desktop design.

# Security

## Authentication and Authorization

Authentication for this system is a factor that must be considered. The entire system is based around alumni users being able to create profiles and log into the system to be able to write about themselves for other alumni or for school faculty to be able to see. There must be two primary places where the user is authenticated. They must be authenticated when creating an account, because they need to be located in active directory in the school’s data, and they need to then be authenticated again when logging in, making sure they have the right password, things like that.

The system must also consider user authorization. Because there’s multiple levels of users in this system, you need to make sure that an alumni user can’t access a faculty level profile or a system administrator’s profile. The same goes for a faculty member not being able to log in as a system administrator, etc. So for example, if an alumni enters the URL for a page only a system administrator can access, then the alumni student will be redirector to their own page or thrown an HTTP error stating that they’re not authorized to access that content.

## Encryption

The system must consider encryption as a part of it’s security features. The system has implemented a messaging service for alumni to be able to contact each other, so because these messages must be private, the system should use some sort of encryption to send these messages from one user to the other. That way a potential attacker listening in on the network wouldn’t be able to see messages going through or listen in on a conversation. There must also be data hashing implemented in the system for storing user passwords. While this is different from encryption, it’s something that must still be considered.

## Confidentiality and Privacy

The confidentiality of users is something that must be considered in any system, however since this system gives open access to all information about a user to other users in the system anyways, confidentiality isn’t something huge that the system needs to worry about. The system is made to help put people out there. Since the information people put out into the system is accessible by anyone registered in the system, there isn’t that much privacy to disclose either. The system should prevent anyone outside the system from seeing user information, but that’s about the extent of it.

## Database security

Database security should always be a concern to a system. Having an unsecured database and giving people external to the system a high chance of managing to manipulate, corrupt or even delete any data would be detrimental to the system. The system should absolutely have database security in it.

## Directory manipulation

Because the system is going to be implemented in .NET, manipulating the directory such as using path truncation or reverse directory transversal shouldn’t be a concern to us as developers. The .NET framework will handle taking care of attacks like that for us, knowing how far back a user can step in a directory, and knowing where to redirect a user to should they attempt to access something they shouldn’t be able to. So, directory manipulation shouldn’t be of a concern because we’re using .NET.

## Injection attacks

Injection attacks such as cross-site scripting and SQL injection are very definetely things that we need to be vigilant about. Between chat rooms for users to be able to communicate, input fields for users to enter data, and all sorts of other places where an attacker could potentially attempt to inject executable code, we want to have all of our input sanitized. We don’t want out database getting tampered with by attackers because we left fields in the registration form vulnerable.

# Defects

Writing a proper bug report is a very important part of testing a system. Writing a proper bug report ensures that the person who’s going to fix that bug knows exactly what’s going on, how it happens, and how to make it happen so that they can try to figure out what’s going on in it. A proper bug report should have a title that gives the reader a very clear idea of either what the bug does, or where the bug exists in the system. Following that, the bug should have a very detailed description of what the bug is. This should include as much detail as possible for the bug, what happened when it occurred, what you we’re doing prior to it occurring, everything. A bug report should also contain an in depth set of steps to reproduce the bug. Everything that someone would have to do in order to get the bug to occur again. The report should also contain a severity and a priority for the bug. These are two very different things. The severity of the bug is along the lines of, does the bug cause the system to crash? Or is it just a typo? Whereas the priority of the bug is how soon does this need to be fixed?

For example, if a bug is found in the system where the system crashes entirely whenever a new user attempts to register for the system, this is a huge bug. The system is crashing in front of the user, which should never happen, so this bug is very severe. However, how often is the system going to get new people registering for it? Does the system get large influxes of registrations after graduation, but otherwise very few a year? If so, then it’s the beginning of winter, we aren’t likely to get people registering right now, so maybe it can wait until next sprint or we just need to make sure it’s fixed before the end of this one. But if we get new people registering for the system on a weekly basis, then this bug should absolutely get fixed as soon as possible.

Another example would be if a little bit of the college news gets cutoff the edge of the screen when the user is on Internet Explorer. This is a very minor bug in terms of severity. A little bit of content not showing up isn’t the end of the world when it’s an issue that only exists in a deprecated browser on a page that people aren’t likely to visit frequently. This bug should be fixed before the end of the sprint if possible, but otherwise it can wait until next sprint without much penalty. Unless the system has an abnormally large amount of Internet Explorer users, this isn’t a huge issue. If this bug existed on Google Chrome, then it’s priority would be increased quite a bit and it should be fixed before the end of the sprint.

# Exploratory Testing

Exploratory testing is very valuable for a system because it allows you to go in without any sort of plan and just try to break stuff. It gives you free reign over what you’re doing, and you only report things when you find a bug, meaning that writing formal test cases for it isn’t something that you generally do. The issue with this kind of testing is that it can often end up wasting quite a bit of time. Because you don’t have a formal test plan, you aren’t going into the system with a real idea of what you’re going to be testing, meaning you can end up messing around on a couple pages for a very long time before ever finding any bugs.

For our system, I would recommend doing some exploratory testing in earlier sprints especially. Because my recommendation for determining our team’s velocity was to underestimate how much our team can do. If we end up with extra time at the end of some of our earlier sprints, we can do exploratory testing to see where we have issues, and then in later sprints we can determine if we want to continue with this type of testing depending on how well the team did previously in it. If they proved to be good at it and it wasn’t a waste of time, then it might be worth including as tasks, but otherwise we just do it in earlier sprints.

# Black Box Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Condition | Input | Expected result | Pass |
| 1. | The system displays a list of x users, where x is the number of users to display selected | 25 | 25 users will be displayed | V |
| 2. | The system displays a list of users matching the search criteria | Year: 2016  Program: science | All graduates from the science program in 2016 will be displayed | V |
| 3. | The system outputs an error to the user because they entered invalid data | Year: ksjdflsajfd | Please enter a valid year | I |
| 4. | The system output an error telling the user to select a valid program. | Program: Something non-existent | Please enter a valid program name. | I |
| 5. | The system has no alumni in record matching the search criteria | Year: 2017  Last Name: Dumaresq | There are no students matching the search criteria | V |
| 6. | There are no search criteria that have been entered | N/A | Please enter a search criterion | I |
| 7. | User clicks to fill in the advanced search criteria | Button click on advanced | The advanced search criteria form will be displayed | V |
| 8. | User tampers with the data and enters and invalid university | University: Something invalid | Please enter a valid university. | I |
| 9. | Alumni student can send a private message to another alumnus | “Hello….” | Other alumni will receive a notification for their message and be able to see what they sent and reply to the message | V |
| 10. | When registering, the password must be at least 8 characters long | 123456789 | The system will accept this password | V |
| 11. | Alumni must enter a valid email address | Phdumaresq | System will display an error message asking for a valid email | I |
| 12. | The information that the alumni enters must match the data for that alumni that the school has on profile. | First Name: Something  Last Name: Invalid  Year Graduated: 1902  Student ID: 01010101  Date of birth: 01/01/2010 | You are not an alumnus of Heritage College. You must be an alumni to register for this system. | I |

# System Testing

|  |  |  |
| --- | --- | --- |
| ID | Test case | Roles involved |
| 1. | User can search the database for alumni matching a search criterion. | System Administrators, System managers, Alumni, Teachers |
| 2. | Alumni can create a profile with basic personal information filled out. | Alumni |
| 3. | Alumni can decide what information they enter and what is visible to people searching | Alumni |
| 4. | Alumni can identify themselves as willing volunteers for events | Alumni, Teachers |
| 5. | Users can look at upcoming events for the college | Alumni, teachers, coordinators, admin |
| 6. | Alumni can send messages to other alumni within the system, as well as see messages that have been sent to them from others. They can choose whether other alumni are able to use the messaging system to contact them directly. | Alumni |
| 7. | The system administrator has full access to be able to see who has logged into the system recently, new users who have registered as well as who has changed their profile recently. | System Administrator, System Managers, |
| 8. | The system Administrators can send mass emails to all the alumni of a certain program. | System administrator, Alumni |
| 9. | System managers can do everything the system administrators can do, as well as modify all the alumni in the system, including adding new ones. | System Managers, System Administrators, Teacher, Alumni |
| 10. | System managers can grant new rights to other users of the system | System Managers, System Administrators, Teachers, Alumni |

# White Box Testing

## Static

Static white box testing methods are tests where you examine the code without actually executing it. Reading over the code, making sure that it makes sense and auditing it. Static white box testing is something that I would not include in the testing for the alumni system. Static white box is long, and can be very easy to make mistakes by misinterpreting code. So static white box testing is something that I would exclude from this project, however I would still include white box auditing of the features of the system. Not to make sure that they work, but to make sure that they all exist. Going through the code, making sure everything is there, looking at how tightly coupled different layers are in the code, making sure things make sense and the whole thing isn’t spaghetti. While this isn’t testing to see if the code works, it ensures that the system will be easier to maintain and continue to develop in the future.

## dynamic

Dynamic white box testing is another very important part of white box testing. Dynamic testing is when you actually execute the code, and watch what happens, see if for a given input, you get your expected output. For dynamic testing, I would include mostly unit tests to ensure that all of our business logic layer code is working. With unit tests, we can come up with edge cases in our code to make sure that our code continues to work as expected after all the changes that we make to it. Unit tests are able to cover almost all code that we write, with the exception of UI facing code, such as what will be output in our Views.

## code coverage

When talking about code coverage, there’s a lot of different things to consider, such as the different types of coverage. Two types of code coverage are statement coverage, and path coverage. Statement coverage is a type of coverage that aims to cover every single line of code in a program. It will include any number of true statements to aim for 100% code coverage to make sure that everything works. This differs slightly from path coverage, where it will aim to cover each possible path through the code, but not necessarily focusing on the decisions through the path. Statement coverage will test true conditions, whereas path coverage doesn’t care about what a decision’s output will be, just that it tests the code that will get executed because of the decision.

# Installation Conversion Strategy

For the installation conversion strategy for this project, we will be going for a parallel conversion style, since not all alumni will be on this system and there’s no way to force them on to it. There’s no real system in place that’s been described as is, so presumably it was just teacher’s with a list of students who don’t mind helping out with stuff that they could email. While those teachers can prompt alumni to register for this system, there’s no way to get them to do it, so the conversion style will likely remain parallel.

The conversion location will be simultaneous. There is no current electronic system in place that we are replacing, just a paper and email system, so all locations will be installed at the same time, which is likely just going to be on the one csprod sever. So with a single install, it kind of ends up being simultaneous by default.

The conversion modules will contain the whole system. Since the system will come out of development as a whole system that’s been tested through and through, there should be no issue pushing this into production as the whole system. There may be some initial release bugs that will need to get smoothed out, but the whole system will get pushed out at the same time.

# Conclusion

In conclusion, the purpose of this document was to discuss the development of the Alumni system for Heritage College. The report has covered different types of project duration estimation techniques, and has recommended which techniques should be used for this system. Following that, the report then went into the velocity of our development team and how we will figure out what our velocity will be as a team. After that, we got in to discussion of writing actual code by deciding on standards that our code will follow. This means code conventions that will be followed as well as styles that will be implemented in the user interface design. Included in this portion was a short paragraph on why coding conventions are important, why we should use them and why they *will* be followed by this team. Following that we got into testing the software, starting off with security. It covered the primary security concerns with this system and how the system should be designed to handle the important issues. Following that was a section on defects and how you will write a proper bug report so that the testing can be effectively done. However, before writing a bug report, we need to know where to test and what types of testing will be done. So, after proper bug reports were discussed, different testing methods that this system will implement was covered, including black box and white box testing techniques, and system testing. Finally, the report concluded with a section on the installation conversion strategy that we will implement and why this is important for the Alumni system.