**Timeline Creation Guidelines**

**Cornell Cup USA – Arm Enabled**

**Prof. David R. Schneider, Cornell Director of MEng Studies in Systems Engineering, drs44@cornell.edu**

Welcome to the Timeline Creation Guidelines. These guidelines are broken into several smaller, more digestible steps for you and your team to follow to create a timeline for your project. It is worthwhile to read through the entire set of steps before beginning as once you understand these steps you may find it better for you to change the order. These are only guidelines to help you get started and not miss any important aspects or value of creating a timeline as this is a critical skill to being a successful engineer.

**Creating Your First Timeline:**

**Step 0.** The first rule in doing these is the same for the first time you do anything: Don’t Panic. In developing a schedule there are a lot of things to think about and it can be difficult to know where to start. Just remember that ultimately, what you are building for yourself is a tool that will help you.

**Step 0 is Complete When:** You and your team are ready to treat this as an important activity that will help you in your professional career.

**Step 1.** Try to define some key milestones. A milestone can have several formal definitions depending on the context, but in short a milestone represents a key aspect of the project with a clear start and ending point. Often a milestone is also defined in terms of achieving a certain kind of functionality within your project, for example: enabling your device to run via battery power, or adding a new feature into a GUI. Other times milestones are created for the work that goes behind making an important decision such as selecting the main microprocessor. Still other times milestones are defined in terms of a subset of required skills, for example: constructing the housing for a suite of sensors; each sensor may require its own special attachment mechanism mounting but the creation of all the sensor attachments are grouped as a milestone because they all require the same mechanical design skills. Often milestones of this later type are created because the work all needs to be accomplished in roughly the same time period or it makes it easier to assign a single individual that set of work. However not all milestones have to be the same length and the amount of work required from one milestone to the next can vary significantly.

You probably won’t get all of the milestones your team needs to accomplish in your first attempt to write them down but that’s okay and is honestly almost always the case. As you continue to develop your timeline you will most likely find that you need to add milestones and change ones that you originally had. **Do not be afraid to do so**, especially in this early stage, but at least this gives you a starting point and can help you start to discuss the project’s work as a team.

**Step 1 is Complete When:** you have a list of milestones that will take your team from the beginning of the project (today) until the entire project, including testing and documentation, is complete. You also should record the time it takes for each one of these timeline creation steps for future planning reference.

**Step 2:** For each milestone, create a list of tasks that need to be achieved in order to accomplish that milestone. Milestone tasks commonly take the sequence of Establish Requirements, Design Feature, Build Feature, Test Feature, and Document Feature. Other common tasks of that form are Re-Iterate Feature, Review Feature, Order Components, Integrate Sub-Features, Feature Interface Testing, Troubleshoot Feature, but there are no rules as to what to call a task. Examine the sample timeline for a small example of the kind of task variety that can occur, but notice that many of the tasks can be thought to fall within the categories above.

**Step 2 is Complete When:** you have a list of tasks for every milestone. Notice that you will start to assume that certain tasks have been accomplished in previous milestones in order for various tasks to begin in later milestones. Make note of these presumed tasks. If warranted add a task to the previous milestone to spell out that this presumed task needs to occur.

It is strongly suggested that different team members attempt to write out the tasks for different milestones and then try to combine them into one complete list in a later in Step 4. When working on things separately however, it is very important that everyone writes down any of the tasks assumptions, especially if you believe that tasks are needed from milestones other teammates are working on.

**Step 3:** Review your list of tasks to see if any subtasks should be listed out. Usually subtasks are a good way to record the events/decisions/tests etc, that must occur in order for a task to be considered complete. Even though you may feel a subtask is implied within a task, (the “do I really need to list that” feeling) listing out a subtask can help make sure it’s not forgotten and will be addressed seriously at a certain point in the schedule. This can be particularly important if you expect someone else will be carrying out the tasks you are creating. It is worthwhile to write out subtasks even for tasks you intend to perform yourself, not only for the practice of creating these tasks but then experiencing what it is like to follow the task lists you have created. You are welcome to create as deep of a hierarchy as you feel is warranted, but at the same point creating subtasks for the purpose of creating subtasks can be counterproductive as well.

**Step 3 is Complete When:** you have added subtasks to your lists of milestone tasks such that you believe that someone else could follow that task list and come out with the same end product you would if you worked on that milestone.

**Step 4:** Combine all of your team’s milestones, including their tasks and subtasks, into one complete list. The order of the milestones should roughly resemble the order in which your team would imagine they could be executed. However at this point it is not worth arguing which milestone should definitely come before which other milestone as some milestones may have tasks that overlap or others may be able to run in parallel. The purpose of this step is really more to review each others’ milestones and tasks and to help make each other be aware of the tasks that you anticipated would be done in other people’s milestones. This may cause tasks to be moved between milestones or milestones to be changed completely.

**Step 4 is Complete When:** you have a single list of all milestones with supporting tasks for the entire team.

**Step 5:** Review the milestone & task list one last time to make sure you have included tasks for testing, ordering parts, creating documentation, and reviewing progress with your advisor(s).

**Step 5 is Complete When:** your team agrees that these tasks have been included where they are needed.

**Step 6:** Take a break. Go have dinner, go to sleep, or even just work on something completely different. You’ve accomplished a good deal here. You will have the opportunity to iterate again on the list while performing future steps but it is more productive to take some time in between iterations.

**Step 6 is Complete When:** your mind is no longer cluttered by the entire list you have created.

**Step 7:** Add a deliverable column to your timeline. For each of the task, subtasks, sub-subtasks, etc list out the deliverables for that task. The deliverables should be the results from a task that once obtained, clearly state that the task has been completed. Examples of a few common deliverables are requirement lists, physical pieces of equipment, test results, written documentation, review feedback, and software functionality. In most cases deliverables are something tangible, even if it’s just a written list, however, a deliverable could also be a key decision. Notice that throughout this step by step guide, there has been a task set up (i.e. Step #) and a deliverable that defines when it is complete (i.e. Step # is Complete When:). Use this same idea and examine the sample timeline as well when creating your deliverables.

**Step 7 is Complete When:** all task, subtasks, sub-subtasks, etc have at least 1 deliverable. Up until now you have kept an idea of what the deliverables of each of the tasks would be. Now it must be recorded and you have definitely defined the scope of what will be accomplished within each task.

**Step 8:** Add columns for and assign ID abbreviations to all milestones, and ID numbers to all tasks and subtasks. By giving every milestone, task, and subtask a unique identifier, it will be easier to refer to these items as you continue your development. Notice the way that they are done in the sample timeline as an easy way to add new subtasks without having to change as many ID numbers. As an example for how to read ID numbers written like this, the sub task “Final Advisor Check” under the “Performance Measures” milestone has the ID: “PM.4.2”.

**Step 8 is Complete When:** all milestones are given ID abbreviations and all tasks and subtasks are given ID numbers. You may however use whatever ID system you feel that will work best for you instead.

**Step 9:** Identify dependencies between tasks. Overall it is generally assumed that when walking down the list of tasks in a milestone that the first task listed must be completed before the second task listed, the second task completed before the third and so on. However this is not always the case. Some tasks may be started early or done in parallel with other tasks. The start of some tasks may even depend on the deliverables of several other tasks to be completed and it is important to note these dependencies in the timeline. Please read rows 2-4 of the sample timeline and the “Resolving the Chicken and the Egg” Scenario below for more details.

**Step 9 is Complete When:** a dependency column has been added to the timeline and the connections have been made between the tasks who depend upon other tasks’ deliverables.

**Resolving the “Chicken and the Egg” Scenario**: When attempting to set the dependencies it may at first appear that you find that two tasks, task A and task B, depend upon each other cyclically, (i.e. like the chicken & the egg). If this is the case split at least one of the tasks into 2, for example let’s split task A into say A1 and A2, such that the first half of A, A1, delivers the part of A that B needs and then task B delivers to task A2 what task A originally needed from B. This may sometime take the form that task A1 is to estimate certain parameters that B needs and then task A2 is to finalize those parameters based upon the results of task B.

A good example is within the Design Demand Algorithm milestone. The algorithm needs to know as much information as possible about its incoming filtered sesnor data, however the Design Sensor Filter needs to know how "well filtered" the data needs to be for the algorithm to run so that they can design the filter appropriately.

As issues like this are often a point of debate between teammates, consider the following 2 scenarios of a discussion between 2 Team Members where one has agreed to take on programming the demand algorithm and the other has agreed to take on designing the sensor filter.

*EXAMPLE SCENARIO 1:*

*Programming Team Member*

Dude, I can’t do anything until your sensor stuff is complete.

*Filter Developing Team Member*

Well I can’t do anything until you tell me what your program can handle. Then I’ll know what the filter has to do.

*Programming Team Member*

But how am I supposed to tell you what you what the program can handle because I haven’t written it yet

*Filter Developing Team Member*

So write it

*Programming Team Member*

But I can’t until I know your input information. It says right here in the book that you need to know about the variance of the incoming data before you can chose which method is right. I think the book knows more than you.

*Filter Developing Team Member*

Oh really. That book doesn’t have the first thing about how to build a sensor filter. It must just take the same stupid stance that you can just build any filter you want.

*Programming Team Member*

Obviously there must be some way to do it. We’re not the first people to ever have filtered sensor data fed into a computer program. Are we? Are we?

*Filter Developing Team Member*

Well what will happen is I’ll come up with something and then you’ll say its no good, and then I’ll have to do it over again and you’ll still say it’s no good and I don’t want to keep doing the same thing over and over and over again if we don’t have to

*Programming Team Member*

Well you have too …you’re just lazy.

*Filter Developing Team Member*

Well you’re just incompetent that you can’t figure your part out first.

*Programming Team Member*

That’s it. You’re off the team!

*Filter Developing Team Member*

No you’re off the team! And dead to me.

*Programming Team Member*

I never want to speak to you again! …except in chemistry because I need your help…

*EXAMPLE SCENARIO 2:*

*Programming Team Member*

Well I know the algorithm is expecting to take this kind of information but I need to know information about the variability of the sensor input before I can decide what is the best method to follow.

*Filter Developing Team Member*

Well I would imagine that the sensor data often needs a high pass filter but aside from that I really don’t know what we’ll need until I take a look at the raw input and it will be a while before the IR encoder is even built.

*Programming Team Member*

Okay let me take a look at what I've read about these algorithms and see if I can find any examples that are close to our situation that might give me a ball park idea

*Filter Developing Team Member*

Sounds good and I’ll look to see if I can find any information on the noise of the components for the IR encoder

Later On…

*Programming Team Member*

I found out that there is a parameter you can adjust as to how much variance you're expecting and the algorithm will be able to compensate. However there is a limit and the reference I found says it can really vary from case to case, so I'm just not sure how much variance the program can handle.

*Filter Developing Team Member*

Hmmm… The components that we might select could vary greatly, so it turns out I really don’t know how much variance the final filtered data is going to have until I create the filter. But I think once we have at least the data from the COTS encoder I can give a worst case estimate.

*Programming Team Member*

In the meantime I know at least most of the data structures will be the same regardless of which method we choose so I can code those up.

*Filter Developing Team Member*

And I’ll help to speed up getting the COTS encoder data.

*Programming Team Member*

Okay then I can use that worst case estimate in the code for starters and see how well it does or how easily it fails. If it does well we might not even need that filtering.

*Filter Developing Team Member*

Great, less work for me.

*Programming Team Member*

But if it fails, hopefully we can identify why it fails…

*Filter Developing Team Member*

…and that can give me the requirements I need for the sensor filter.

*Programming Team Member*

Perfect.

*Filter Developing Team Member*

And looking at the timeline and the other dependencies we have, it looks like I can use the time that I am waiting for that data to begin work on the energy storage system.

*Programming Team Member*

You’re right, that will be a very effective use of our time.

*Filter Developing Team Member*

Have I ever told you that I love you.

*Programming Team Member*

OMG, I love you too!

*Filter Developing Team Member*

I’m so glad we made the timeline!

And they lived happily ever after.

Chances are most of the time you will experience something in between these two. Watch yourself for phrases similar to those above. Although they’re exaggerated here to make a point, if roadblocks like this start to occur, this is a good warning sign that you should re-examine your planning. Focus on what you CAN do with what you have at moment, not what can't do, and think about what can be done to help you progress toward your goals.

**Step 10:** Mark the task(s) within each milestone that can be started before any other task has to be completed with an asterix, a.k.a. the milestone’s initial or entry tasks. Any task that does not depend on any other task in the milestone is an entry task. In the sample timeline, the asterix is placed in the dependency column.

**Step 10 is Complete When:** Every milestone has at least one identified entry task, and all entry tasks have been identified and marked with an asterix (or similar marking of your choosing).

**Step 11:** Add an Effort Column. Estimate the amount of effort that you think each task would take. Often rather than thinking of things in terms of days, it can be more helpful to first try to rate the effort of various tasks relative to each other. Days often can be a difficult effort measure, particularly as a student when your day to day schedule can vary significantly. A common way to do this effort rating is by ranking each task on a 1-5 or 1-7 scale (a 1-10 scale is actually considered too variable for most situations). Effort is also commonly used rather than a strict time-to-complete estimate because different people (or subteams) may work at different rates.

If you have sub-sub-sub tasks it is not necessary to give all of these sub-sub-subtasks a separate effort rating, but instead just let the parent task have an effort rating that is representative of all of the subtasks combined. Similarly, if you have a task that you feel should have a very large effort rating, it is typically best to try to split the task into several smaller tasks.

**Step 11 is Complete When:** All tasks, subtasks, etc have an effort rating associated with them. Each team member should assign efforts to the tasks that they believe they could take on. Then all of the effort values are compiled into the one master timeline. It is important to look over each other’s efforts level to try to make sure that your effort ratings are consistent, but consistency is easier to achieve often after Step 12.

**Step 11a: (optional)** This step targets computer programming but the ideas here can be applied to any task. Computer scientists (and many of the developers of this competition consider themselves as computer scientists as well) have the unfortunate reputation of being some of the worst at estimating the amount of time it takes to complete a task (one of the reasons being “feature creep” which will be talked about later). One of the most successful ways to help deal with this is to have more depth in the subtasks and more detailed deliverables.

As an example, consider the task “Program Data Structures and General Architecture”. This may sound like a straightforward task and hence may receive a lower to medium effort level. However when this is detailed out, the amount of coding can be quite substantial albeit not necessarily the most thought-intensive or challenging aspect of the project.

For every class there would have to be fields, accessors, get & set methods, copy methods, clear methods, constructors & destructors methods, all of which have to be created. This is assuming, of course that the class architecture has already been decided upon and no significant changes are determined during the implementation phase.

In addition more complex data structures may require additional methods; lists may require special sort methods, trees special balancing methods, and hash tables proper key generation just to name a few. Furthermore, the more complex the data structures, the more internal documentation that will need to be written.

Now imagine that there was a task to create a new GUI functionality; another common task that is often assigned less effort than actually required. On top of everything mentioned above, there would have to be update methods, draw methods, input handling methods. Then on top of that there would finally be the methods that create the actual functionality that is desired, and unless pseudo-code has already determined for the algorithms that need to be program, it is easy to be over confident or miss some of the “housekeeping” or interface elements that may be required to implement the algorithm.

Hopefully the task “Determine Architecture & Data Structures” will help to better inform the effort required for the “Program Data Structures and General Architecture” task however it can be very easy underestimate the “standard” elements of a task. Just because you are confident that you know how to do something doesn’t mean that it still won’t take a lot of time.

The best way to handle this is to take the time to spell out everything that can be done. Then do your best to try to keep track of how much time you have spent doing each subtask so you can review it to improve estimates in the future. In some cases it can even be helpful to track how long it takes to do specific activities. For example, for programming it can even be worthwhile to keep track of how much time you spend planning your code, writing your first version, how long it then takes until it compiles, how long you spend debugging it, and how long you spend writing documentation. If nothing else, detailing out additional subtasks and deliverables can be very helpful for communicating to your teammates or future boss how much work that you actually do.

An activity that has been consistently highly rated by students is to keep an even more detailed log of your time spent throughout the entire day. Make a log book (or file) that you keep on you at all times. Keep track of how much time you spend in class, on reading assignments, written assignments, in group meetings, surfing the web, in extracurriculars, doing leisure activities, eating, sleeping, home chores, etc. If you do keep a log it is recommended that you are diligent in doing so for at least 2 weeks in order to notice important trends. This can be hard to force yourself to do, but ask your advisor if they are willing to collect this material as part of your grade. The value that past students have said this activity has had for them in improving their time management and the fact that it counts towards your grade, just might give you the added incentive you need to make sure you treat this activity seriously.

**Step 12:** Convert your effort ratings into hours of work estimates. Doing the effort rating of Step 11 prior to assigning times however can lead to far better end results regardless of whether you are experienced at creating timelines or not.

**Step 12 is Complete When:** a Time Required column is added and filled in on the spreadsheet. Again, each team member should convert the efforts for the tasks that they believe they could take on. Then all of the time values are compiled into the one master timeline. It is important to look over each other’s time values to try to make sure they are consistent.

**Step 13:** Take another break. Remember it is often the most challenging when we ask ourselves to think about solving problems in a brand new way, particularly the first time when have yet to experience the results of that work in action. Learning to define a challenge and then define a potential solution that you can be confident in to meets the challenge’s needs is a new experience for many students. Then working out the tasks necessary to create that solution and verify that solution meets those needs is another new challenge. But these challenges are critical to developing yourself as a professional engineer and developing skills that are highly sought after by potential employers. This project gives you the opportunity to demonstrate proven experience to potential employers that you can take your ideas to reality. These skills will serve you well especially when working in teams and even more so when working in larger teams from very different backgrounds, as you will be doing in your future careers. But for now take another break. You earned it.

**Step 13 is Complete When:** You are again refreshed and ready take on the next steps. It is natural for this process to take some time and by taking a break it can help you review what you have already accomplished.

**Step 14:** Even the best engineers and many successful engineer managers have found that, especially for engineers who are just starting their careers, that in order to gain a more accurate estimate of the total number of hours required, the managers take the time estimates from their team and then multiply those estimates by 2.5 in order to determine the actual time for completion and it is the 2.5x time that they report to their clients. (If you followed step 11a closely you may not need to multiply this by quite as much.) Unfortunately many engineers discover this during the execution and this can lead to schedules falling behind and a lot of late nights in the end. If there are late nights in the end it should be because you have already accomplished everything that you promised in your proposal and you are examining additional optional features.

**Step 14 is Complete When:** you have reviewed your goals, you have made any final adjustments to your time-required schedule, and you have added an actual time spent column (aka Time Actual column) to your timeline. Especially in the beginning of your project, do not be too hard on yourself if your tasks take longer than you expect. But the important thing is that you observe why they have taken longer and learn to either adjust your time required estimations in the future or learn to avoid or minimize the observed time sinks in the future. For your own sanity and well-being, in your weekly life, plan for your work on projects to take 2.5 times as long as your estimate. Then in the worst case you’re left with extra time on your hands.

**Step 15:** Note tasks’ requirements for special equipment/resources & their availability. This may not be an issue for all projects but can be very important for others. If you haven’t done so already, add tasks to your timeline that have to do with obtaining or reserving time with special equipment/resources. Then add a dependency to the tasks that require the actual use of the special equipment/resources. Tasks that represent the returning of the special equipment/resources can also be helpful in certain cases as having the return date being the due date of the task can be a helpful way to maintain this important information within your timeline.

Again, as it can be harder to determine the specific tasks later in the timeline, it can be hard to determine these tasks needs as well. Therefore recognizing the need for special equipment/resources early on, and adding these kind of tasks to the timeline, will be something that you will have to watch closely for throughout the project.

**Step 15 is Complete When:** Tasks have been added to the timeline to represent the dependency of current tasks on obtaining special equipment/resources. Overall anything that is out of your direct control that could influence your ability to progress on a task can be added to the timeline at this point as a new task and dependency.

**Step 16: (optional)** Convert the tasks into a PERT or Gantt Chart. Depending on the software you have created the timeline in, this will be performed differently. If you are doing this by a spreadsheet, these charts may have to be made by hand. However, if you are doing it by hand you can group those tasks that have no dependencies as a single node in the charts. If doing this by hand you may also find it helpful to start at the end of your project and work your way backwards following the dependencies you’ve already listed.

The creation of these charts and paths is outside of the scope of this guide. However these charts can be worthwhile in helping you establishing what is your “critical path”. A critical path in a project plan is an ordered sequence of tasks, linked by dependencies, such that if any one of those tasks is delayed, the end project date must also be pushed back. Knowing what this sequence of tasks is can greatly help you recognize which tasks must be started as soon as possible. Determining the critical path can also help you determine how much any task can be delayed without delaying the entire project; this is also referred to as the task’s “slack”. You are encouraged to look up more on this, especially if your advisor thinks its worthwhile.

**Step 16 is Complete When:** The PERT or Gantt Charts have been created with the dependencies and task durations entered. All that would be required to create an ideal “earliest start schedule” is a starting date.

**Step 17:** Examine your schedule for bottlenecks. A bottleneck is a point within the schedule where it is difficult for any progress to be made until a very small group of tasks or even a singular task is completed, i.e. that task or small group of tasks is called the bottleneck and could hold up the rest of the project if it is not completed on time. If you determined your critical path earlier, it can be easier to identify potential bottlenecks. Regardless of whether you have or not, look for tasks that many other tasks depend on (these are especially easier to see if you have created a PERT chart). It can be helpful to add an additional optional column, a Deliverable To column, to your timeline that lists which other tasks are dependent on this task being completed. If a task shows up repeatedly in the Deliverable To column, there is a good chance that that task is a potential bottleneck.

These are not the only potential source of bottlenecks. However being aware of the dependencies and looking for bottlenecks throughout your execution of your timeline can help you make smart decisions early on to either reassign more people to an important task, leaving a less time critical task for later, or reassign people towards tasks that can be done in parallel. Also look for how you can do everything you can possibly do towards the tasks relying on the bottleneck task such that when that bottleneck deliverable is available (i.e. that one part you ordered finally came in) you are instantly ready to make use of that deliveriable. (i.e. all of the supporting wiring and physical attachments for that part have already been made, you just need to plug that part in and you can begin testing)

Unexpected bottlenecks can occur and may be outside of your control. A late part, a piece of equipment breaks, unexpected results, or even additional assignments from an outside class can cause a part of the project to lag. If you can look for bottlenecks ahead of time, you can be prepared to make the adjustments more logically and early on before the bottleneck has had a serious impact on your overall schedule.

**Step 17 is Complete When:** You have identified likely tasks as being bottlenecks and which tasks need to be done prior to prevent potential bottleneck issues. It is recommended that you highlight or otherwise signify any time critical tasks within your timeline and similarly highlight in a different way what tasks/actions could be done should the time critical tasks become a bottleneck issue.

**Step 18:** Group milestones and tasks into sets of goals for your team to accomplish within given periods of time. Common periods of time to set as goals for projects like this one are either weekly or bi-weekly. As students you are naturally trained to think of assignments in terms of 1 week periods. However, bi-weekly project goals often work well with student life because if you know you’ll have a heavy week with your normal classes, you can plan to make up for it during the other week and still meet your overall 2 week goal on time. Neither is truly better than another and it really depends on what works best for your group. Some teams even function better by having goals set every 3-4 days for themselves. Sometimes it’s also better to have shorter goal periods when you are working on aspects that require significant integration and cooperation, and sometimes longer goal periods when work can be done more independently.

**Step 18 is Complete When:** You have goal time periods set for at least what will be your current goal and what will be your goal time periods for at least the next month, where all of the tasks’ deliverables in those goal time periods are well defined. At the end of this step, you should also have a rough estimate when all of your milestones will be complete and hence a rough estimate for when the final project will be complete.

**Step 19:** Just before or after (or both) you have scheduled to complete a goal, add a task to review your work over the past goal time period. This task should be used to check the progress on all tasks, update the timeline, and confirm with everyone on the team, including your advisor, what you plan to accomplish during the next goal. Often it is helpful to discuss what will be accomplished within the next 2-3 goal time periods as well as review where you are in terms of the overall project.

**Step 19 is Complete When:** the goal review tasks are entered into the timeline. As far as the effort and time to devote to these review tasks, a guideline to how to structure a typical 1 hr review meeting and what to cover is provided for you below.

* 15 Minutes to go over recent:
  + Accomplishments & Demonstrate any new features
  + Describe any Issues encountered during the latest goal period
  + Cover any Remaining Concerns & Challenges
* 1-5 Minutes to update on your progress with regards to the overall project
* 10-14 Minutes to go over the Next Goal and any Timeline Updates

Leaving 30 minutes for discussion with your advisor as needed throughout. Include in the goal review task the amount of time it would take to prepare the materials necessary for the above topics so that you can be as efficient as possible in your meeting.

**Execution of your Timeline:**

Your timeline is now complete enough to be a very useful tool. Ultimately the purpose of making these lists is to create for your team an easy step by step task list so you can check your overall and individual progress and to help identify problems/setbacks/bottlenecks early on.

Despite all of your hardest efforts to do so, it’s okay if your actual work does not perfectly fit your timeline. You are students and this is part of the learning process. What is CRITICAL though is that you continually update your timeline and review how your estimates and the actual time spent match. Keep a saved version of your timeline after completing every major goal you have set, and be prepared to submit with your final report intermediate versions of your time; a suggested frequency between versions submitted may be 1 updated version of your timeline per month of the project.

Scoring your performance on your timeline will be partly on the quality of the documents you create as a tool for your team but also on your team’s use of the timeline as a tool throughout your project. Being able to following your timeline well is fantastic, but for this project learning from your mistakes and improving in your ability to estimate, plan, and adjust is just as great. Overall please be objectively honest in your assessment of your own work and speak with your advisor(s) about your use of this tool as they are there to help you and they will also be providing comments on your progress as well.

You may begin working towards your project and following your timeline anytime when you are ready, even before the application has been submitted.

**Beware of “Feature Creep”:**

Projects like these can be very exciting. You have remarkable control over the direction of your work and it will succeed by the energy and effort you put into it. It is common however, in the development of one feature, that the opportunity to create additional new features or improve upon an existing feature may arise.

With the freedom you have in a project like this, these sudden realizations of opportunity can appear especially attractive and motivational. However it is important to be aware that these ideas can both be outstandingly beneficial or potentially severely crippling by taking away time from other parts of the project.

So what should you do? First of all, jot it down. Regardless of whether you pursue the idea now or not, get the flash of potential brilliance recorded in some fashion. Once you have the initial idea in its entirety written down, treat the idea as a small milestone; determine tasks, subtasks, deliverables, resources & effort required, etc. It may feel like it takes a little wind out of your sails to formalize the idea, but if that “loss of wind” occurs it is more often due to the realization of just how much extra effort the new ideas will take.

This is far better than venturing headstrong down a certain path and then realize midstream that the end is far more complicated than you realized. Often the initial feeling of "This is so cool! I just gotta do this to make it happen" leads to another "well I just gotta do this now and then it will be done" which can lead to a "well it’s taking a bit longer than expected but once it’s done it will be great" and then to "well it will take me at least one more week but then I can go back to those other things I was working on". This is often the core of the feature creep problem and how exploring new ideas that initially seem very exciting may take away from other project aspects that ***need*** to get done. Instead the new idea would be better addressed later on in the project or perhaps left out entirely until the next iteration of the entire project.

This is not to stifle creativity by any means! It is just to help you realize how you to focus your creativity into achieving what is best for the creating the very best overall project. Good ideas will never truly die, they just come back stronger. If now is not the right time, the idea will come back and you’ll be stronger then to. Just be prepared that the comeback may not happen right away. Even if it’s not until the next project or even years later, you have it recorded, and when the time comes you’ll be able to pick it up again.

Then again, sometimes when the new idea is right for the time you’re at now, the process of determining its required tasks can help keep the momentum alive or even grow because you realize your idea can be achieved now and with great benefit to the project.