

Software Project Plan

Individual Portion

Kevin Roberts

Tasks:

I will be doing backend stuff some weeks and GUI stuff other weeks. We have now decided who will be doing what for the entire project.

For the first checkpoint, I will be doing GUI stuff, making a very simple interface which will simply display a window with a button that says "Sync", as well as text with the status of whether it is in the process of syncing or if the sync is complete.

For the second checkpoint, I will be doing backend stuff, working on the updating of packages already on the system (basically an emerge -uD world).

For the third checkpoint, I will be doing backend stuff again, on the browse functionality, which means grabbing the list of all of the packages and storing them in an N-ary tree to be passed up to the frontend.

For the fourth checkpoint, I will be working on the backend install functionality. Which given a package structure, will call emerge and install it.

For the fifth checkpoint, I will be working on a transparent sync, we have decided to automatically sync before doing an update world to get around the problems of scheduling a sync at some point in time, with perhaps the user shutting their computer off when not in use and so never getting the syncing done.

For the sixth checkpoint I'll be doing lots of things, making the package list skip any fetch-restricted packages, working on the backend configuration file to specify how often and when to run automatic updates, if indeed the user wants to, as well as working on the Graphical frontend to allow the user to easily customize when they want the program to auto-update.

For checkpoint 7, I will be working with everyone else to do any testing, bug-fixing, documentation, etc, needed to get this all working for the final submission.

If I have extra time during any of the checkpoints, I will be working on a reverse-dependency check for uninstalling packages that will allow for any dependencies required by this package that aren't required for any other package currently installed to be uninstalled as well. This is not the top priority by any means, but as a personal goal, it would be cool to get this done.

Time Estimates:

Week 1- Emerge Sync GUI: 3 hours. I have gone through most of the GTK tutorial, so I'm not too worried about having to spend a lot of time learning GTK. However, I will be having to fork off a separate process to do the integration with the backend, so that may take some time to figure out, and there are some other issues with the communication that may need to be resolved. Also, I will be team programming with Aaron, and I have always worked solo before and I'm unsure about the dynamics there so I'm allowing some extra time. I've spent a little bit over 3 hours total so far on the most common word program, including documentation and learning maps, so it really shouldn't take me 3 hours to do this tiny little program even with the forking of processes which is new, I'm just allowing extra time for the team element.

Week 2 – Emerge -uD world and sync GUI: 2 hours. The backend forking and execing should already be in place from the last week's project, as well as getting the output from the other process, so the biggest part of this project would be parsing the emerge -uD world output to put together a list of packages to return, as well as determining if there are blocks or fetch restrictions. The partner dynamics will change as I'll be working with Chris on the backend this time, still allowing some extra time for that, but not as much as hopefully I'll have adjusted some.

After spending much more time than anticipated on the first two checkpoints, not because getting stuff working took longer, which it didn't. But, rather because I spent a lot of time ironing things out, working on things that other people were having difficulty with, reworking some of my code to make it more modular and easier to work with, in particular, I got things to work by using fork and creating a new process for the GUI, but I was using global variables, and all of the recommended uses of fork seemed to indicate that you should then run another program in the child process rather than using it for two processes within the same program, p_threads were the recommended way to go for what I was looking for, so I reworked it using p_threads, got rid of the global variables, made it very nice to work with and will save a bunch of headaches later on as we have more things in the GUI requiring separate processes.

Week 3 – Backend browse: 6 hours. I'll be working with GNodes which I've never worked with before, I'll have to work with esearch which the commander has not been made yet, although Chris also needs this for his as well, so he'll probably be doing most of the work on that as the rest of his work is very little.

Week 4 – Backend install: 1 hour. This is mostly completed due to the emerge world functionality done before, the integration may still take some time, but primarily I'll probably be using the additional time this week to work on the reverse dependency problem, or else working ahead on the next checkpoint.

Week 5 – Transparent Sync: 1 hour. Again, this is mostly completed since we have the sync functionality, we have the emerge world functionality, we just have to make it so that one automatically follows the other, I'll probably be working on the reverse dependency stuff quite a bit this week as well, or else some on the auto-update stuff for next week.

Week 6 – Skip fetch-restricted packages, auto-update frontend and configuration: 7 hours. There may be some new GUI widgets to choose the time, or days, or such of when the updates will be happening, the exact options the user will have to choose when to auto-update has not been decided yet, so I'm giving myself plenty of time for this one.

Week 7 – testing, documentation, bug fixing: 10 hours. No way to really anticipate how long this one will take, but I imagine it will be quite a bit, especially if I'm still working on the reverse dependencies.

Dependencies:

There are no dependencies for the first part of the first project, but the integration of the GUI part I am working on with the backend part other people are working on will require that the backend be completed. Once the integration is done, there will be no more dependencies for the next project, until we get again to the integration. The way we have our schedule mapped out with discrete checkpoints with frequent integration of backend and frontend, it is nearly always the case that each week we need to get the frontend and backend done, we need to then integrate before moving on the next step. There are some things that can be worked on before previous stuff is done, but not always.

Unknowns

As I mentioned earlier, I have already taken the steps to learn GTK, and even though I still have a lot to learn, the first hurdle is over and I'm not too worried about more advanced stuff I may have to do.

Also mentioned earlier is the forking and execing, Jason has already worked this stuff out and I'll just have to look at his code and utilize that, so nothing much to worry about there I don't think.

Reverse Dependencies: is this possible? I think so, but I could be wrong. Is it practical? Will it simply take too much time to determine all of this that it isn't even worth it? I'll be working on making it not take long to determine, my hope is that even if takes some time to go

through and gather all of the data, it won't take long to check the reverse dependencies of any particular package. The regathering of data can just be done after an update.