CS31310 Worksheet 3: Major Project Process

Outline of Project

The title for my major project is "visualisation and topological aspects of high dimensional data". This is a research orientated project which will most likely involve looking at dimensionality reduction and manifold learning algorithms, applying them synthetic data where the reduced dimensionality of the data is known and to real life samples and comparing the results. We will most likely be using cancer research data as the real life samples.

Process Description

Defining a process for this project is difficult for two reasons. Firstly, the major project is an individual endeavour which means that many of the practices used in software development methodologies which involve multiple people are not applicable (e.g. paired programming from XP). Secondly, being a research orientated project, it is difficult to define exactly what the end result of the project should be. These factors make the choice of specific methodology for the project non trivial and I therefore propose cherry picking the best concepts from different methodologies to form an appropriate process suitable for the major project.

However, there are several factors that immediately appear to be several aspects that would be desirable in any methodology used for the major project. The first is that such a process must be flexible. In a research orientated project the exact final outcome of project is unlikely to be specifically known at the start. An ideal process should be open to modification as the project progresses. There must be scope to change the initial design if some ideas do not pan out.

Another requirement is that regular progress should be shown to be made. My supervisor will probably be concerned about the speed of progress if I spent the first half of the project time gathering requirements and designing the system with nothing tangible to show for it. The processes therefore should include some form of metric for measuring progress.

It is also desirable that the process incorporates some kind of quality checking. The end result of this project will most likely be some kind of processing pipeline for analysis. It is therefore important that there is some confidence that each component in the pipeline is working correctly and that further modifications do not break the system. This is important so that we can be confident that the results of a complicated system are not just rubbish.

From these initial thoughts it immediately becomes obvious to me that the processed used for the project should take most of its ideas from the agile end of the development methodology spectrum. From the extreme programming methodology I would take the concept of planning games and short iterations which would allow me to design and develop the project by decomposing it into different stories which could be completed in a relatively short amount of time. This gives both a concrete set list of things "to-do" on the project while also being flexiable; stories can be added or dropped as required and design can wait until work on a particular story begins. Using stories and iterations for development also provides a metric by which to measure progress (i.e. the number of stories completed in an iteration).

From extreme programming I would also utilise the take the concepts refactoring and test driven development. These two components would allows me to be flexible in the way I design the system but in a controlled way that prevents me from "just hacking it together". Refactoring would allow me to produce a design on the fly and testing would give me confidence to make those changes.

I would also import the concepts of YAGNI and general simplicity from XP. I think that the right approach to this project will be to write something simple first and evaluate the performance of the system then modify the design of (or add features too, see last paragraph) the system based on these results.

Finally, I would also take the concept of continuous integration. This concept is less important in an individual project where there is only one contributor, but would still be helpful. This would allow me to work on different features in parallel without fear of breaking system as a whole when merging changes into the large system.

Discussion

Each of the major methodologies used in software engineering have their strengths and weaknesses and there are many different aspects that I would like to incorporate into a methodology for the major project. However, features from different methodologies cannot be selected at random. Aspects of development methodologies often support one another and to include one principle in designing a process requires another to be included as well. The most obvious example that comes to mind is with an agile approach. Including refactoring into the process is desirable, but in order to be able to refactor effectively the developer needs confidence that they won't break the system. In agile this confidence is given by TDD. It is essential that complementary features are included together to avoid the issue of hacking it together.

As I have mentioned in the preceding section strict engineering methodologies such as the waterfall methodology are not appropriate for the major project. The fact that the project is a individual effort that is unlikely to be safety critical removes the need to use such a rigid process. It is inflexible and requires far too much big design up front in a project where the end goal isn't certain. The only potentially useful feature of the waterfall methodology is that it includes definitive cut off points that show progress. However, this can be achieved in an agile approach by using frequent releases.

In the preceding section I have suggested that the extreme programming methodology would be the correct one to use for my major project, however I do think that there are some other aspects from other methodologies that could be utilised in the project. For example, while I don't necessarily support creating a big design up front, I do think that the high level walkthrough involved in the initial stage of a feature driven approach would be useful in planning the major pieces of work to be completed during the project. To echo what I said I first worksheet I believe that performing a small amount of design up front, or at the start of the particular story is not a bad thing, providing it is open to change and refactoring later on.

The real main reason why I suggest XP for the major project is the flexibility. The project topic is deliberately designed to be open ended and flexible. Depending on my own tastes and interests as well as what results we see as the project progress will influence the end result of the project. Therefore I think that it is essential to use a methodology that can adapt to change. I also think having short iterations that implement a small amount of stories will be important. Having an iteration period of a week would be beneficial because I am likely to meet with my supervisor at least once during this period. It would probably be at this point where a "release" of the project would be created. This would also provide a logical point at which to plan the next iteration's stories.

In agile terms I think that my supervisor is going to most closely resemble the customer. While it is not practical for my supervisor to be truely "on-site", they should at least eb fairly accessible. I will be meeting with them at least once a week, possibly more, and will probably have regular email contact with them. This also suggests that agile will be a good fit. I will need to work closely with my supervisor and potentially with other members of staff throughout the project to ensure that I am on the right track and building the correct thing.

In summary, I think that an agile, XP based approach is the correct choice of methodology for my major project. The small size of the project and team coupled with the importance of flexibility and regular, meaningful progress and the accessibility of the "customer" suggest that this would be an appropriate choice. Using such an approach will prevent me from falling into the trap of "just hacking" it together. I think that some thoughtful design will be required up front before rushing in a implementing stories, but this can be relatively informal. Stories should help me to focus me attention on specific parts of the system during weekly iterations, ensuring I make meaningful progress.