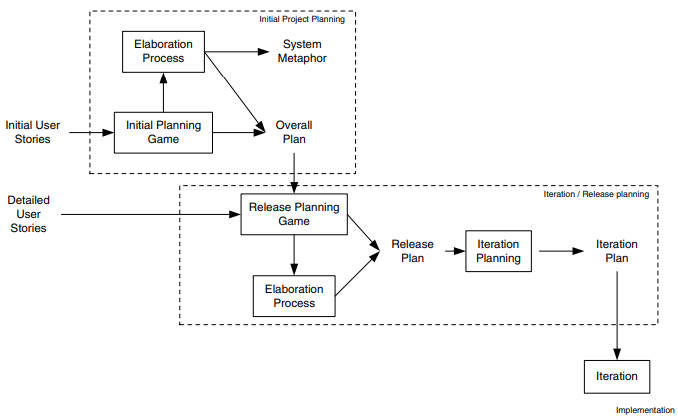
* **Agile modelling and planning XP projects**
* Here we will consider how and where Agile Modelling fits into the project-planning aspects of an XP project
* An XP project is planned at a number of levels and at various points during an XP projects lifetime
* This means that Agile Modelling practices may be more or less relevant at different stages during this process.
* In the following, we will consider the planning process and where Agile Modelling can be exploited to the benefit of XP





* **Initial Project Planning**
* There are two primary steps within the initial project-planning phase; these are the initial planning game and the elaboration process.
* *Initial planning game*. During this process, business and development may resort to modelling to help them clarify the user stories. By applying Agile Modelling practises, this modelling can be controlled and focused. An example of where they might do this is when a User Interface mock up might be created, with some simple flowcharts to prototype system behaviour as a way of elaborating a user story. For example, Figure illustrates a possible user interface design for a membership web site. This is a diagram drawn on a white board to consider what fields are needed and what will happen when a user selects the submit option. The flow diagram presented in Figure expresses what happens when the submit button is pressed. Again, this is a diagram drawn on a white board.
* *Elaboration process*. During the elaboration process, various models maybe created to help the developers understand what will be required of the system. This will help to produce better estimates, etc. Again Agile Modelling practises can be of great help here.
* **Iteration/Release Planning**
* During the iteration/release planning stages, modelling is again important.
  + ***Release planning game***. As with the initial planning game, Agile Modelling practices can help focus the modelling activities used to clarify user requirements.
  + ***Elaboration process***. Although this is typically a shorter process than for the initial project planning phase, some modelling often still takes place and Agile Modelling can be applied to ensure that modelling does not become a burden.
  + ***Iteration planning***. In order to break down user stories into tasks, it may be necessary to model how the user stories might be implemented. This might involve initial class structures, behavior, etc. This can allow tasks to be identified, clarified or split up. Note that this is not large up-front design, as the models may be discarded and may only be intended to help elaborate the tasks.
* **XP implementation phase**





* This is where the code actually gets written within an XP project. There are therefore various points at which a model may be relevant and therefore Agile Modelling practises may be applied.
* For example, in helping to understand code in order to refactor it, etc. We will now look at how Agile Modelling can complement several of the implementation-oriented practises of XP.
* Implementation of oriented practises, mean practises such as “Test-first coding,” and “Refactoring, “rather than the more process-oriented ones such as, “The Planning Game” or the“40 hour week rule.”
* The practises to be looked at in this section are:
  + refactoring,
  + test-first coding,
  + simple design and
  + Pair programming.
* **Refactoring**
* Refactoring is primarily a code improvement technique, so is it compatible with a modelling activity? Is modelling and Agile Modelling in particular, relevant to refactoring? The answer of course is “yes,” as we have already indicated earlier in this chapter. In the last chapter, some of the issues to consider when refactoring were given as:
  + Make sure you know how to improve the code.
  + Make sure what you have done has improved the code.
* That is, “know what you are doing!” It was also stressed that you should not refactor: “When you haven’t got a clear plan of how you will improve the code.”
* Agile Modelling can be used to help with all three issues. By modelling various aspects of the system, you may gain a better understanding of what it does.
* By modifying the model, you can evaluate how you might refactor it and whether it appears to have benefited from the refactoring.
* This is a lot cheaper than actually coding the changes and then considering the results. It may also be a better medium through which to convey your ideas to your pair-programming partner or to others; thus allowing improved communication of ideas.
* By applying Agile Modelling practises, you can determine if they do actually need to be revised or not. For example, by considering the following two Agile Modelling practises:
  + Update only when it hurts not to do so.
  + Discard temporary models.
* You might decide not to update the existing models until absolutely necessary. Even then, I find it better to wait until someone shouts for the models and update them at that point (in Just-In-Time fashion) as an XP project may refactor the code several times before the point at which someone wants to reference the documentation. This would lead to unnecessary model revisions taking place.
* **Test-First Coding**
* At first sight, with respect to test-first coding, modelling may seem at best superfluous and at worst contradictory. This is because, in test-first coding, you essentially follow this cycle:

1. Write a test.

2. Write the code to be tested.

3. Run the test/get the code to work.

4. If the test has passed, then return to step 1 until finished.

* So where does Agile Modelling fit into this cycle? It may fit in at a couple of points. You may have decided to carry out a small amount of modelling at the start of the current task in order to understand what you need to do.
* If you have taken into account the “Design for Testability” Agile Modelling practise, then it may have helped to identify the tests to be implemented. Another point at which Agile Modelling may be relevant is once a test has been written and you need to consider how to implement the business code.
* A short “stand up” design session with your pair-programming partner can be invaluable. It can help to clarify any number of issues and help to make sure that they (and you) are engaged in what is being done and that no one is just along for the ride. This also fits with the Agile Modelling practise “Prove it with Code.”
* Thus, if you modify the test-first coding cycle to the following, then you are maximizing this XP practice, as well as, supporting the Agile Modelling principle of rapid feedback:

1. Write a test.

2. Model the solution.

3. Implement the solution.

4. Run the test/get the code to work.

5. Discard temporary models.

6. If the test has passed, then return to step 1 until finished.

* **4 rules of Thumb**
* Given all of what has been said above, what can we say about where XP and Agile Modelling should be applied? There are actually a number of rules of thumb that can be identified. Note that these are rules of thumb and are not hard and fast rules. For example, XP can be and has been applied successfully to larger projects. However, it is more difficult and there is a greater potential for failure, particularly for those inexperienced in agile approaches. The rules of thumb that have emerged include:

1. Smaller projects of typically less than 10 people. The larger the project, the harder it is to manage as a pure XP project. The XP approach, that is to a large extent self- organising, becomes difficult with 20 or 30 developers involved. Of course, the team can be broken down into smaller groups and treated as smaller XP projects, but then that’s what you have, smaller XP projects interacting.
2. Known domain/applications. For larger projects, XP projects work best where the domain and the type of application are well understood.
3. Well-established architectures. This point is related to rule 2 in that the reason that XP works well in well-understood applications is that there is a (possibly) implicit architecture. The developers know what they should do where, when and how. If this is not the case, then an architecture needs to be established within which the XP project can operate.
4. Scalability not an issue. If scalability is an issue, it must be considered early on in the project so that it does not become an issue later on. This is typically a problem in larger, longer-lived projects where it is difficult to see the scalability issues early on. Again, an architecture may help.