**Methodologies Overview**

**Standard Waterfall:**

The waterfall model follows a simple, sequential development process where each phase depends on the deliverables of the previous one. This traditional approach requires detailed and precise up-front planning and documentation before any work is started. This means that all the user’s requirements must be analysed and validated until there is a straightforward, clear path and end-goal to the project. The name ‘waterfall’ stems from how the development process mainly flows in one direction, downwards, like a waterfall, through the phases, conception, through analysis, design, implementation, testing and finally maintenance. Each stage of development proceeds in a strict order.

Advantages:

Due to the waterfall model involving a lot of documentation defining each phase in the project, new developers can be quickly brought up to speed at any stage during the development process. The strict, rigid procedures also enforce discipline among the team so that their sole focus is on the current task. This makes large teams easier to manage for project managers as each stage has a specific deliverables and milestones that need to be kept on top of. In addition, during the analysis, the costs can be estimated with a fairly high degree of accuracy and so the budget could be changed or planned for at an earlier stage.

Disadvantages:

Although the waterfall model is very robust, it has several disadvantages, for instance, it cannot accommodate changing environments and requirements. This method is only suitable for projects where the requirements are fixed and well-known. Another major drawback of the waterfall approach is the limited involvement of the end-user or the customer which increases the risk of customer dissatisfaction of the end product as they cannot give feedback during the design and implementation stages which may lead to redevelopment increasing the cost and time. Another critic is that the project management is very repressing, and this does not help encourage creativity within the team and use their skills in the best way. As the package is only integrated at the end of the lifecycle, often called a ‘big bang’, issues are likely to surface with incompatible code between developers due to miscommunication, resulting in redevelopment increasing time and money.

**Rapid Application Development (RAD):**

Rapid Application Development is a type of incremental and iterative development model which was designed in response to the critiques of the waterfall model. RAD compresses several stages of the waterfall model into an iterative process that prioritises prototyping and a higher level of user involvement and feedback. The development team delivers a fully functional prototype of the product which is then met with an evaluation from the user or customer. The prototype is then refined to the user’s needs and this process repeats until the prototype evolves into the final working product that will satisfy the user/customer. This is known as evolutionary prototyping, however, RAD also uses throw-away prototyping where instead of refining the existing prototype, a new version of the software package is created from the ground up accommodating the new needs of the user.

Advantages:

One major advantage to RAD is that there is less up-front planning meaning more time can be spent on developing and testing the software product. Additionally, the user requirements do not need to be well defined in the beginning as prototyping takes care of this. RAD divides the project into several compartments for which separate prototypes can be developed in parallel, speeding up the development process and shortening the life-cycle. Changing environments and requirements can be easily incorporated in the evolving prototypes and so this reduces the risk of not meeting the customer’s requests and therefore leads to a higher customer satisfaction.

Disadvantages:

RAD is inapplicable to projects with low budgets as the cost of Computer-Aided Software Engineering (CASE) tools for modelling and automated code generation are very high. Additionally, RAD requires highly skilled developers and designers who have lots of experience and a large skill set in order to swiftly respond to the customer’s changing requests. The RAD model also demands more frequent user involvement throughout the development lifecycle which would require a lot of commitment from the customer to achieve the targeted prototype in the given time frame.

**Scrum:**

Scrum is a lightweight process framework and a subset of Agile methodologies. Scrum uses iterative and incremental practices with a main goal of increasing productivity and reducing the time taken to complete the project. It is designed for teams of up to nine to twelve developers who break their work into milestones that are completed in time-boxed iterations called sprints. These sprints last no more than two to four weeks and at the end of each sprint, a fully integrated and tested product that is potentially releasable must be released for the customer to review. At the start of each day during a sprint, a Daily Scrum (or a Stand-Up meeting) is held at the same time and same place and is restricted to 15 minutes. During the Daily Scrum, each Team member explains what they completed yesterday, what they plan to complete today and if they have any issues that will stop them from completing their tasks.

Advantages:

Scrum is very efficient at enabling development teams to complete project deliverables in a rapid fashion and this is one of the main reasons that Scrum is the most popular Agile methodology, according to the [12th](http://stateofagile.versionone.com/) annual State of Agile report, 70% of software teams use Scrum or a Scrum hybrid. Scrum is extremely suitable for large projects as it breaks it up into manageable sprints for the development team. The Team in Scrum are self-organised and choose amongst themselves which parts each member will tackle in alignment with their specific skill-set. This improves efficiency and ultimately creates a software package that is better in quality. User feedback is an important aspect of Scrum, being an Agile model, and so this ensures that the final shippable product is in line with the user’s standard and satisfaction.

Disadvantages:

For Scrum to work at its best, all individuals must be fully dedicated and cooperative, including the customer and/or user. A consistent high level of communication between all participants through each backlog and review is necessary for this framework effectively. If this is not the case, the project is bound to fail. In addition, Scrum requires highly experienced software developers and designers to reduce the risk of scope creep, which is when the project’s scope grows uncontrollably at a point after the process has begun. Poor definition of tasks and requirements can also lead to failure as it leads to inaccuracies during the development process.

**Dynamic Systems Development Method (DSDM):**

Dynamic Systems Development Method is another agile project delivery framework that focuses on the full project lifecycle, using an iterative and incremental approach with continuous user/customer involvement. It makes steps to ensure the feasibility and business sense of a project before it is created. DSDM emphasises on in-budget and on time product delivery with variable functionality to address projects of all sizes. In order to keep to the budget and be on time, DSDM uses MoSCoW prioritisation to leave out unnecessary requirements, reducing the project scope. The MoSCoW mnemonic simply means Must haves, Should haves, Could haves and Won’t haves. DSDM makes a heavy use of prototyping and this ensures that all interested parties have a clear picture of the various aspects of the system.

Advantages:

In DSDM the basic functionality is delivered quickly with more functionality being delivered at regular time-boxed intervals in the form of the evolving prototype, this means the customer can see evidence of progress being made from very early on in the process. Like all agile methodologies, the constant user feedback helps ensure that the system being developed meets their needs requirements. As DSDM, unlike other methodologies, focuses on fixed time and resources, the system is always delivered on time and budget making it very reliable for the customer to invest in. Additionally, due to the early integration and testing in each increment, also known as Integrated Testing, hidden flaws will surface early in the process reducing the risk of redevelopment. MoSCoW prioritisation helps to eliminate requirements that are deemed unnecessary and would only waste time and money.

Disadvantages:

DSDM focuses more on delivering the product within the proposed timescale, this can often lead to decreases in code robustness which could further lead to integrating issues later. Like other agile frameworks, DSDM requires full commitment from all individuals, even the user/customer, involved in the process for smooth and successful development. In addition, highly-skilled teams are essential in both the business and technical areas to ensure efficient workflow as DSDM is often a complex development process.

**Extreme Programming (XP):**

Extreme Programming is another subclass of the agile development framework and is specifically designed for software development projects. XP is distinguished for its immense focus on adaptability rather than on predictability. This is because of fluid nature of software development in general where requirements cannot be fully predicted from the beginning and will constantly changing as the project moves on. XP consists of very little to no documentation during the process and instead XP programmers start to generate code from the very beginning while listening to the customer. As a result, XP developers need to be naturally good listeners to enable them to understand and develop solutions which match the customers’ needs and desires as close as possible. XP emphasises on the testing the function of the system and so tests are pre-written and then the application code is development to pass those initial challenges of the test, further tests are also written to try and break the code.

Advantages:

XP is extremely fast-paced and so instead of taking a couple of years to complete a project, like in waterfall, XP projects only lasts for several months. This is because there is continuous integration and testing rather than a ‘big bang’ in the waterfall approach. Additionally, there is no wasting time writing lots of documentation that are often never used or seen again, further reducing the development lifecycle. This also benefits the developers as they will be much happier writing code rather than writing documentation leading to them being more creative during the design and development process. In XP, changing requirements are swiftly implemented throughout the development period, this reduces costs as it reduces the risk of a complete redevelopment.

Disadvantages:

XP consists of a very fast-paced work environment and so demands for a highly motivated, stable and experienced development team who can rapidly translate the customers’ words into a working software package. XP takes customer/user involvement to the extreme as they must always be on-site to answer questions and ensure that the software being developed is in line with their wishes. Due to little documentation in XP, constant changes are not kept track of, causing unexpected failures to occur and even when bugs are resolved, without accurate records, same errors may recur. XP can also be very stressful as developers have to keep to tight deadlines. This stress may cause them to make more mistakes, diminishing the quality of the final product leaving an unsatisfied user/customer.

**My Conclusion:**

I have chosen to pursue the waterfall approach for my project, as I feel it is the most suitable for designing and developing a new management system for JD Dog Care. JD Dog Care already has an existing manual system in place and so the Waterfall methodology is the most appropriate to deal with implementing a manual system into a software one. The Waterfall model is easy to understand and follow for developers with minimal experience and I can work more efficiently when I know what I am doing. JD Dog Care has clear, well defined and mostly unchanging requirements for its software system and reliability in the system is somewhat critical for John and Diane to ensure smooth operations in their ever-growing business.