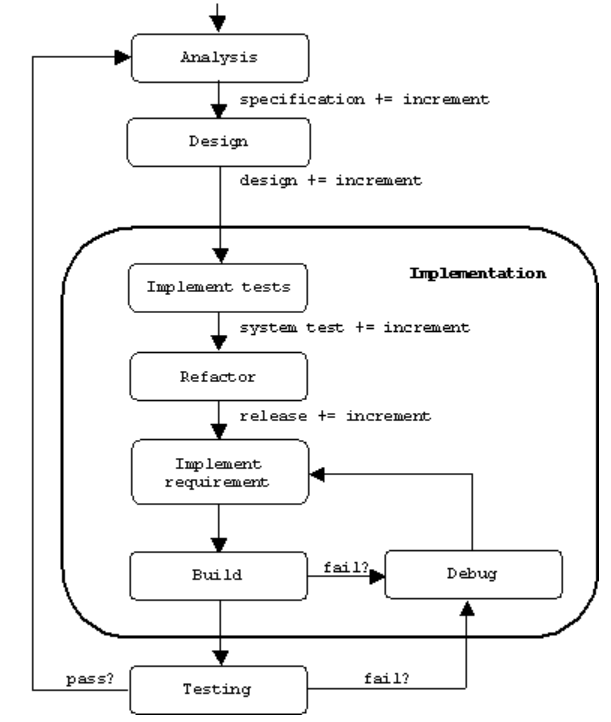
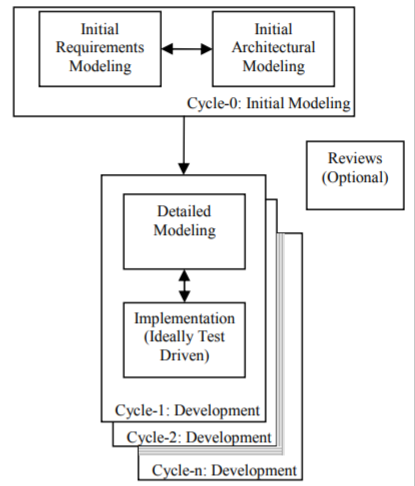
**The Agile Model Driven Development**

In this part, I will introduce Agile Model Driven Development (AMDD) that builds on the MDD and is also the agile type of the MDD. It is worth mentioning that the Agile Model has been a common model for the development of software product, because the iteration development of the AMDD is short-term (usually 1-3 weeks for one iteration) that divides the entire task into smaller segments, and it is used as an incremental process. The following figure can represent the iteration process of AMDD.



*Fig: iteration process of AMDD [1]*

Compared to the MDD, the goal of the Agile Models for AMDD is to be “good enough” rather than a large number of models. From the above figure, we can draw the conclusion: AMDD is an evolutionary method. The modelling process refers to the analysis and design steps, followed by the code implementation. In these processes, the iteration can be executed according to the requirement. AMMD emphasizes the design. To deal with the iteration, using the refactoring to improve and update the design.



*Fig: AMDD project level [2]*

The AMDD project level that is known as the first process of AMDD in the literature shows the process of iteration and a high-level life cycle *[2]*. The objective of cycle-0 activity is to conduct the **initial requirement modelling** and **initial architectural modelling** based on the understanding of specification requirement. The process in cycle-0 – initial requirement modelling is beneficial for stakeholders to participate in the project, the requirement model is regarded as an effective tool to communicate with stakeholders. The other process in cycle-0 – initial architectural modelling refers to make some charts around the architecture. The agile software development can be scaled, the later work will depend on the technique of the initial architectural model. Besides, it can show the architecture and evolutionary design of the system, which can be developed and updated by the iteration. During the development iteration (1,2 … n), developers can take several hours in the **detailed modelling**. It needs more detail and combines these details with models. Followed by the detailed modelling is the several minutes **model storming**. Model storming process – a just in time (JIT) model, stakeholder can draw some sketches to express the examples of their requirement. Developers and stakeholders get together to communicate their thoughts until reaching an agreement. After that, implementation process could take most time in the AMDD process. In general, it adopts the **Test-Driven Development** (TDD) to conduct and code the modelling. TDD in the AMDD is the integration of code refactoring, database refactoring and test. Using the TDD can facilitate the high-quality product due to the improvement of coupling and cohesion. The reason why adopt TDD in the implementation process of AMSS is that Agile Modelling is helpful to the model iterations; while TDD is helpful to programming implementation iterations. Besides, TDD can achieve 100% coverage test to each line of code, which encourages confirmatory validation to the code and detailed specification – it is also the process of agile acceptance tests *[3]*. Code Refactoring can advance the code and design according to the update requirement. Overall, the aim of the project levels of AMDD is to scale agile development, these high level processes achieve analysis, design, code and test with iterative-incremental process.

Next, I will take the use of Agile Modelling in AMDD as an example. From the above, we can see: Agile Modelling (AM) is essential in the AMMD process. It integrates the requirement and principle and is flexible for the iteration process to improve the modelling. In addition, AM is used as a communication tool between developers and stakeholders. It’s simplicity and flexibility are significant benefits. To conduct the AM effectively and take the full advantages of modelling, a series of standards and core principles of AMDD has been established. The detailed core principles which refer the [4] are as follows.

The first primary principle – software is your primary goal. All work in the AMDD aims at facilitating the high-quality software. All design, modelling, documents are devoted to this goal. The second principle – enabling the next effort is your secondary goal, which asks the developers should take into account the future extension. This principle needs the software with high-quality system and the sufficient documents and models to record the information of the software. The next principle - embrace change. Actually, change is a common issue for the agile model driven development. The positive attitude for the team to handle the change and collect the feedback form stakeholders is vital. Besides, this principle corporates with the second principle to support the change by a good preparation. The fourth principle is incremental change. As is mentioned above, change is normal and it should be solved by incremental method rather than solving the whole change together. I hold the view that a high-quality software should be perfected one after another to gain the high-quality software. The fifth principle is travel light. Reasonable number of models and documents is important. AMDD should have the agility and flexibility characters, if there are too much documents, it will become a heavy burden on managing and updating the change. Or a little document, it is difficult to develop a high-quality software. The sixth principle is model with a purpose. Each type of modelling has the specific objectives, pros and cons, knowing about the modelling with purpose can help developers to understand the project well. The seventh principle – assume simplicity. Unlike the MDD, AMDD do not need complex resources, such as modelling. In the majority situation, the easiest approach is equivalent to the best approach. This practice can avoid the time, cost and resource wasting and achieve the high-quality software by the maximum efficiency. Next principle is quality work. Just like its name implies, developers should use the time and investment with high efficiency to improve the progress and quality of the software. The ninth principle is multiple modelling, it is acknowledged that there are various modelling tools and solutions. Choosing the suitable tools and solutions according to the practical conditions. The last principle - maximize stakeholders investment. Developers have to satisfy the need of clients and investors. These principles are core for developers to make the AM successfully, it is also the essential to AMDD process.

**The application of UML in AMDD**

UML is used as the sketch in the process of AMDD, it is usually used during the initial modelling (before the extreme programming) and model storming stages. The reason why UML is important in the AMDD process is that UML can express the requirement of stakeholders so that developers can understand their task. It can represent the interactivity clearly between users. The features of UML 2.0 are suitable for the AMMD. Due to the extensibility of UML, it improves the interface of other OMG modelling prototype on the architecture, using the MetaObject tool to set the language meta model, which simplifies the interaction between tools. Besides, UML is beneficial for the accuracy of models in terms of structure model and behaviour model. Furthermore, it supports the encapsulation and scalability. UML in the AMDD can be profitable for the feedback, the requirement change test feedback can be shown on the this model as soon as possible.

Reasonable use of UML can improve the effectiveness of software development, it should make not only simple model as much as possible, but also make sure achieve the requirement. The general process to make the UML to conduct the Agile Model:

(1) Analysis the requirement and create the basic use case model. This model shows the functional requirement of the system in the object-oriented design approach.

(2) Conduct the initial architectural modelling, it tries to determine a wonderful architecture for the future work. The initial modelling will be gradually developed by the iteration and more understanding on the project.

(3) System design (refining system and detailed design), The flexibility of UML is appropriate for the refining of system. Besides, this stage can write new classes, user interface and improve the database to refine the work on the design stage. In the implementation stage, the UML model is directly converted into the program code.

Overall, UML can deal with the large-scale, complex, constantly changing requirement effectively.

**Conclusion**

In conclusion, our essay pays attention on the MDD and AMDD. The MDD emphasizes the UML models, and the code and test are based on the UML. The code idea of MDD follows “Everything is model”. Based on this core idea, the process of MDD has been introduced. The four stages (requirement definition phase, analysis stage, programming phase and test stage) are represented in details. The development model of MDD from the perspective of metamodel and model mapping technology sharing, model reuse, pipeline development and model to metamodel to illustrate how the MDD-based system stores, manages and publishes the metadata at the application and system layer. In this essay, we introduce the conception of OpenText and Mendix.

This essay illustrates the goals of OpenText. In addition, the OpenText process suite - a set of components that can build the new processes and applications faster and easier. The Mendix can provide software tools, architecture and methods in order to test, deploy and manage. Besides, we introduce the use and advantages of Mendix. Then, we compare the OpenText and Mendix in terms of language, highlights and architecture. AMDD integrates the benefits of Agile and MDD. It has great advantages on the short increment and iteration process. AMDD is differ from the MDD, it does not need too much model, models are the basic for the extreme programming part, TDD and refactoring. In addition, models will be correct constantly by the iteration process. The process of AMDD and its details are illustrated in this essay. It illustrates what is the core principle of AM to achieve a high-quality software in the short term. The use of UML modelling in AMDD is be shown as an example to explain the steps and significance. Owing to the specific advantages, I believe that the AMDD and MDD can be more widely used in the future.

Acknowledge

The AMDD is not a very popular and fully developed area, it is developed in recent years. These website and PDF have explained the principle of AMDD process. Ambler has made lots of contribution and comprehensive explanation on the AMDD.

Reference

[1] <http://www.cs.sjsu.edu/~pearce/oom/se/agile.htm>

[2] <https://orbilu.uni.lu/bitstream/10993/2304/1/Reza-Agile%20Model%20Driven%20Development%20An%20Intelligent%20Compromise.pdf>

[3] <http://agiledata.org/essays/tdd.html>

[4] <http://msoo.pbworks.com/f/Scott+W.+Ambler+-+Agile+Modeling.pdf>