**Aya Mahmoud Abdelaziz Elsebeay**

**Software Enginnering**

### Agile Methods ( ****Dynamic Systems Development Method (DSDM)) ,**** ****Feature-Driven Development (FDD).****

**4CS**

**Dynamic systems development method** (**DSDM**)

is an [agile](https://en.wikipedia.org/wiki/Agile_management) project delivery framework, initially used as a [software development method](https://en.wikipedia.org/wiki/Software_development_methodology). First released in 1994, DSDM originally sought to provide some discipline to the [rapid application development](https://en.wikipedia.org/wiki/Rapid_application_development) (RAD) method. In later versions the DSDM Agile Project Framework was revised and became a generic approach to project management and solution delivery rather than being focused specifically on software development and code creation[[clarification needed](https://en.wikipedia.org/wiki/Wikipedia:Please_clarify)][[citation needed](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)] and could be used for non-IT projects. The DSDM Agile Project Framework covers a wide range of activities across the whole project lifecycle and includes strong foundations and governance, which set it apart from some other Agile methods The DSDM Agile Project Framework is an [iterative and incremental](https://en.wikipedia.org/wiki/Iterative_and_incremental_development) approach that embraces principles of Agile development, including continuous user/customer involvement.

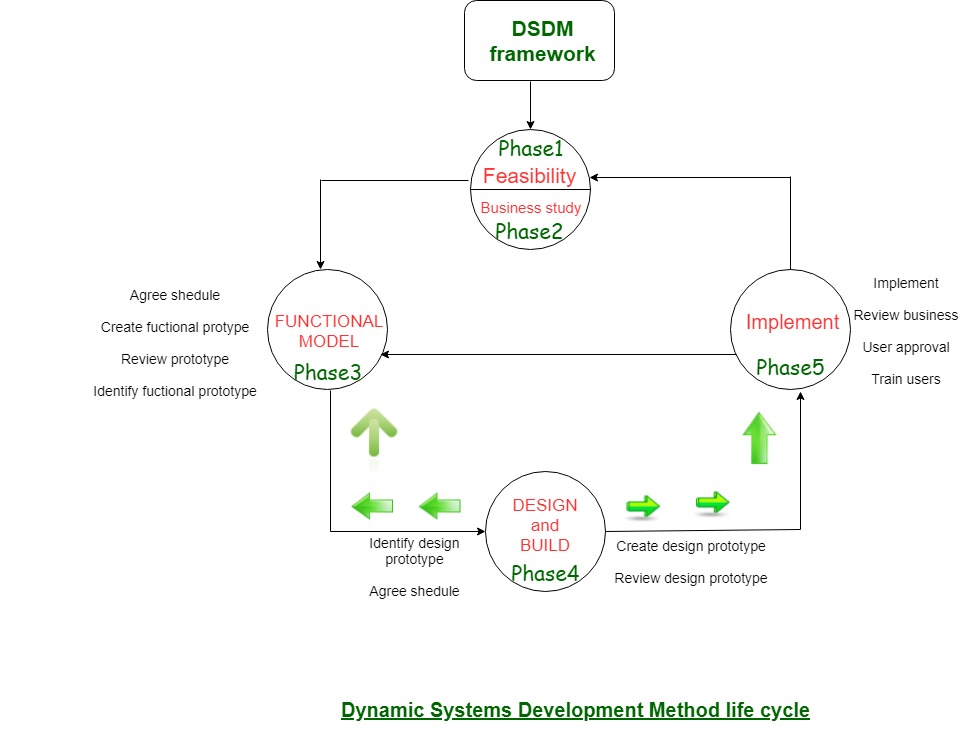
DSDM fixes cost, quality and time at the outset and uses the [MoSCoW prioritisation](https://en.wikipedia.org/wiki/MoSCoW_prioritisation" \o "MoSCoW prioritisation) of scope into musts, shoulds, coulds and will not haves to adjust the project deliverable to meet the stated time constraint. DSDM is one of a number of [agile methods](https://en.wikipedia.org/wiki/Agile_software_development) for developing software and non-IT solutions, and it forms a part of the Agile Alliance.

In 2014, DSDM released the latest version of the method in the 'DSDM Agile Project Framework'. At the same time the new DSDM manual recognised the need to operate alongside other frameworks for service delivery (esp. [ITIL](https://en.wikipedia.org/wiki/ITIL)) [PRINCE2](https://en.wikipedia.org/wiki/PRINCE2), Managing Successful Programmes, and PMI The previous version (DSDM 4.2) had only contained guidance on how to use DSDM with [extreme programming](https://en.wikipedia.org/wiki/Extreme_programming).

**Dynamic Systems Development Method (DSDM)**

The Dynamic Systems Development Method (DSDM) is an [agile framework](https://www.productplan.com/glossary/agile-framework/) that addresses the entire project lifecycle and its impact on the business. Like the broader agile philosophy, DSDM is an iterative approach to software development, and this framework explicitly states “any project must be aligned to clearly defined strategic goals and focus upon early deliver of real benefits to the business.” The framework is built on fourprinciples: feasibility and business study, functional model and prototype iteration, design and build iteration, and implementation

The **Dynamic Systems Development technique (DSDM)**



#### The DSDM Philosophy, Principles, and Project Variables The DSDM Philosophy

The Dynamic Systems Development Method (DSDM) is an [agile framework](https://www.productplan.com/glossary/agile-framework/) that addresses the entire project lifecycle and its impact on the business. Like the broader agile philosophy, DSDM is an iterative approach to software development, and this framework explicitly states “any project must be aligned to clearly defined strategic goals and focus upon early deliver of real benefits to the business.” The framework is built on four principles: feasibility and business study, functional model and prototype iteration, design and build iteration, and implementation The Dynamic Systems Development Method (DSDM) is an [agile framework](https://www.productplan.com/glossary/agile-framework/) that addresses the entire project lifecycle and its impact on the business. Like the broader agile philosophy, DSDM is an iterative approach to software development, and this framework explicitly states “any project must be aligned to clearly defined strategic goals and focus upon early deliver of real benefits to the business.” The framework is built on four principles: feasibility and business study, functional model and prototype iteration, design and build iteration, and implementation

**Feature-Driven Development (FDD)**

**Feature-driven development** (**FDD**) is an [iterative and incremental](https://en.wikipedia.org/wiki/Iterative_and_incremental_development) [software development process](https://en.wikipedia.org/wiki/Software_development_process). It is a [lightweight](https://en.wikipedia.org/wiki/Lightweight_methodology)[[*according to whom?*](https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Words_to_watch#Unsupported_attributions)] or [Agile method](https://en.wikipedia.org/wiki/Agile_software_development) for developing [software](https://en.wikipedia.org/wiki/Software). FDD blends a number of industry-recognized[*[according to whom?](https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Words_to_watch" \l "Unsupported_attributions" \o "Wikipedia:Manual of Style/Words to watch)*] [best practices](https://en.wikipedia.org/wiki/Feature_Driven_Development#Best_practices) into a cohesive whole. These practices are driven from a client-valued functionality ([feature](https://en.wikipedia.org/wiki/Feature_(software_design))) perspective[*[clarification needed](https://en.wikipedia.org/wiki/Wikipedia:Please_clarify" \o "Wikipedia:Please clarify)*]. Its main purpose[*[according to whom?](https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Words_to_watch" \l "Unsupported_attributions" \o "Wikipedia:Manual of Style/Words to watch)*] is to deliver tangible, working software repeatedly in a timely manner in accordance with the Principles behind the [Agile Manifesto](https://en.wikipedia.org/wiki/Agile_Manifesto).

FDD was initially devised by [Jeff De Luca](https://en.wikipedia.org/wiki/Jeff_De_Luca) to meet the specific needs of a

15-month, 50-person software development project at a large [Singapore](https://en.wikipedia.org/wiki/Singapore) bank in 1997. This resulted in a set of five processes that covered the development of an overall model and the listing, planning, design, and building of features. The first process is heavily influenced by [Peter Coad](https://en.wikipedia.org/wiki/Peter_Coad)'s approach to [object modelling](https://en.wikipedia.org/wiki/Object_oriented_design). The second process incorporates Coad's ideas of using a feature list to manage functional requirements and development tasks. The other processes are a result of Jeff De Luca's experience. There have been several implementations of FDD since its successful use on the Singapore project.

The description of FDD was first introduced to the world in Chapter 6 of the book *Java modelling in Color with UML*by Peter Coad, [Eric Lefebvre](https://en.wikipedia.org/w/index.php?title=Eric_Lefebvre&action=edit&redlink=1), and Jeff De Luca in 1999. Later, in Stephen Palmer and [Mac Felsing](https://en.wikipedia.org/w/index.php?title=Mac_Felsing&action=edit&redlink=1)'s book *A Practical Guide to Feature-Driven Development* (published in 2002), a more general description of FDD was given decoupled from Java modelling

Feature Driven Development (FDD) is not as well-known as many other agile frameworks .But when you’re dealing with a large, long-running project, especially in an organization where agile is still mostly confined to [software development](https://www.digite.com/agile/agile-software-development/), FDD may be your friend. Feature Driven Development bridges the gap between traditional controlled waterfall approaches and the emergent processes found in agile approaches like [behavior driven development](https://www.digite.com/agile/behavior-driven-development-bdd/), [extreme programming](https://www.digite.com/agile/extreme-programming-xp/), and [scrum](https://www.digite.com/agile/scrum-methodology/).

**Feature Driven Development (FDD) Practices:**

The designing of FDD happened when the rest of the frameworks were not working for Jeff. This framework is a combination of best practices from other software development frameworks.

We need to understand a few terms like **what is a feature** and **how do we develop by feature** before understanding the practices followed in FDD.

To understand the **feature**, we first need to understand the **function**. The client wants the development team to develop software. Clients would wish to have certain features in the software, and those features will have respective functionalities. These functionalities are known as **Functions**.

In **Feature Driven Development** (FDD), a feature can be developed and delivered to the customer within one or two weeks, depending on the team size and feature complexity.

To make it more clear, let’s consider MS office as software that the customer wants. Now in MS office, the client would wish to have:

* MS word,
* MS Excel,
* PowerPoint

FDD was designed to follow a five-step development process, built largely around discrete “feature” projects. That project lifecycle looks like this:

1. Develop an overall model
2. Build a features list
3. Plan by feature
4. Design by feature
5. Build by feature

Advantages:

* FDD provides the team with a great understanding of what's expected from them and gives them insight into the project's context and scope.
* You don't have to spend your time in meetings. While Scrum uses daily meetings, that's not the case with FDD. They rely on documentation to communicate.
* User-centric approach, and in this case, the client is the end-user.
* FDD is great for large-scale and long-term projects. This framework is scalable, and it can grow as your organization grows.
* With the help of FDD, you can break features into smaller chunks, which will make it easier for you to make a quick turnaround, reduce risk, fix coding errors, and track your progress.

Disadvantages:

* FDD is not suitable for smaller projects and doesn't work for those involving developers only.
* It highly depends on the chief programmer who needs to act as a mentor, lead designer, and coordinator.
* It doesn't offer written documentation to clients, even though plenty of documentation circles among team members.
* It is more focused on individual code ownership.
* It may not work well with other systems.