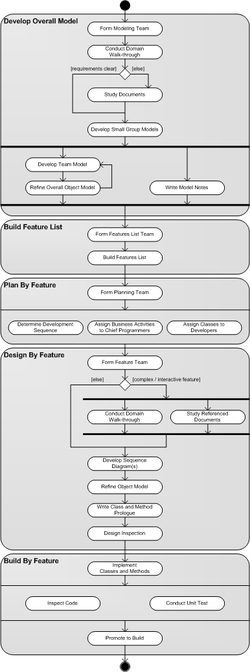
**Feature-driven development** (**FDD**) is an [iterative and incremental](https://en.wikipedia.org/wiki/Iterative_and_incremental_development) [software development process](https://en.wikipedia.org/wiki/Software_development_process). It is a [lightweight](https://en.wikipedia.org/wiki/Lightweight_methodology) 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 [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. Its main purposeis 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 is a model-driven short-iteration process that consists of five basic activities. For accurate state reporting and keeping track of the software development project, [milestones](https://en.wikipedia.org/wiki/Feature-driven_development#Milestones) that mark the progress made on each feature are defined. This section gives a high level overview of the activities. In the figure on the right, the [meta-process model](https://en.wikipedia.org/wiki/Meta-Process_Modeling) for these activities is displayed. During the first two sequential activities, an [overall model](https://en.wikipedia.org/wiki/Feature-driven_development#Develop_overall_model) shape is established. The final three activities are [iterated](https://en.wikipedia.org/wiki/Iteration) for each feature.

[](https://en.wikipedia.org/wiki/File:Fdd_process_diagram.png)

Process model for FDD

**Develop overall model**[[edit](https://en.wikipedia.org/w/index.php?title=Feature-driven_development&action=edit&section=3)]

The FDD project starts with a high-level [walkthrough](https://en.wikipedia.org/wiki/Software_walkthrough) of the scope of the system and its context. Next, detailed domain models are created for each modelling area by small groups and presented for [peer review](https://en.wikipedia.org/wiki/Peer_review). One or more of the proposed models are selected to become the model for each domain area. Domain area models are progressively merged into an overall model.

**Build feature list**[[edit](https://en.wikipedia.org/w/index.php?title=Feature-driven_development&action=edit&section=4)]

Knowledge gathered during the initial modeling is used to identify a list of features by functionally decomposing the domain into subject areas. Subject areas each contain business activities, and the steps within each business activity form the basis for a categorized feature list. Features in this respect are small pieces of client-valued functions expressed in the form "<action> <result> <object>", for example: 'Calculate the total of a sale' or 'Validate the password of a user'. Features should not take more than two weeks to complete, else they should be broken down into smaller pieces.

**Plan by feature**[[edit](https://en.wikipedia.org/w/index.php?title=Feature-driven_development&action=edit&section=5)]

After the feature list is completed, the next step is to produce the development plan and assign ownership of features (or feature sets) as [classes](https://en.wikipedia.org/wiki/Class_(computer_science)) to [programmers](https://en.wikipedia.org/wiki/Programmer).

**Design by feature**[[edit](https://en.wikipedia.org/w/index.php?title=Feature-driven_development&action=edit&section=6)]

A design package is produced for each feature. A chief programmer selects a small group of features that are to be developed within two weeks. Together with the corresponding class owners, the chief programmer works out detailed [sequence diagrams](https://en.wikipedia.org/wiki/Sequence_diagrams) for each feature and refines the overall model. Next, the class and method prologues are written and finally a [design inspection](https://en.wikipedia.org/wiki/Software_inspection) is held.

**Build by feature**[[edit](https://en.wikipedia.org/w/index.php?title=Feature-driven_development&action=edit&section=7)]

After a successful design inspection for each activity to produce a feature is planned, the class owners develop code for their classes. After [unit testing](https://en.wikipedia.org/wiki/Unit_test) and successful [code inspection](https://en.wikipedia.org/wiki/Code_review), the completed feature is promoted to the main build.

Milestones[[edit](https://en.wikipedia.org/w/index.php?title=Feature-driven_development&action=edit&section=8)]

Since features are small, completing a feature is a relatively small task. For accurate state reporting and keeping track of the software development project, it is important to mark the progress made on each feature. FDD therefore defines six milestones per feature that are to be completed sequentially. The first three milestones are completed during the [Design By Feature](https://en.wikipedia.org/wiki/Feature-driven_development#Design_by_feature) activity, and the last three are completed during the [Build By Feature](https://en.wikipedia.org/wiki/Feature-driven_development#Build_by_feature) activity. To track progress, a percentage complete is assigned to each milestone. In the table below the milestones and their completion percentage are shown. At the point that coding begins, a feature is already 44% complete (Domain Walkthrough 1%, Design 40% and Design Inspection 3% = 44%).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table 1: Milestones** | | | | | |
| **Domain Walkthrough** | **Design** | **Design Inspection** | **Code** | **Code Inspection** | **Promote To Build** |
| 1% | 40% | 3% | 45% | 10% | 1% |

Best practices[[edit](https://en.wikipedia.org/w/index.php?title=Feature-driven_development&action=edit&section=9)]

Feature-driven development is built on a core set of [software engineering](https://en.wikipedia.org/wiki/Software_engineering) [best practices](https://en.wikipedia.org/wiki/Best_practices) aimed at a client-valued feature perspective.

* **Domain Object modelling**. Domain Object modeling consists of exploring and explaining the domain of the problem to be solved. The resulting domain object model provides an overall framework in which to add features.
* **Developing by Feature**. Any function that is too complex to be implemented within two weeks is further decomposed into smaller functions until each sub-problem is small enough to be called a feature. This makes it easier to deliver correct functions and to extend or modify the system.
* **Individual Class (Code) Ownership**. Individual class ownership means that distinct pieces or grouping of code are assigned to a single owner. The owner is responsible for the consistency, performance, and conceptual integrity of the class.
* **Feature Teams**. A feature team is a small, dynamically formed team that develops a small activity. Multiple minds are always applied to each design decision, and multiple design options are evaluated before one is chosen.
* **Inspections**. [Inspections](https://en.wikipedia.org/wiki/Software_inspection) are carried out to ensure good quality design and code primarily by the detection of defects.
* **Configuration Management**. Configuration management helps with identifying the source code for all features that have been completed to date and maintaining a history of changes to classes as feature teams enhance them.
* **Regular Builds**. Regular builds ensure there is always an up-to-date system that can be demonstrated to the client and helps highlight integration errors of source code for the features early.
* **Visibility of progress and results**. Managers steer a project using frequent, appropriate, and accurate progress reporting from all levels inside and outside the project based on completed work.