Software Development Methodologies

Lecturer: **Raman Ramsin**

Lecture 10

Agile Methodologies: FDD

Feature-Driven Development (FDD)

1. Introduced by De Luca and Coad in 1999, originally as a tailored complement to the "Object Modeling in Color" technique.
2. A revised version was published by Palmer and Felsing in 2002, which had been completely decoupled from "Modeling in Color", and was general enough to be considered an independent methodology.
3. Based on expressing and realizing the requirements in terms of small user-valued pieces of functionality called *Features*.
4. Not a full-lifecycle methodology:
    1. Starts when feasibility study and overall project planning have already been done, a business case has been established, and permission has been granted by the sponsors to go on with the development.
    2. Excludes post-implementation activities such as system-wide verification and validation, and the ultimate system deployment and maintenance.

FDD: Features

1. Each feature is a relatively fine-grained function of the system expressed in client-valued terms.

2. Conforming to the general template:

   <action> <result> <object>

   for example: "*calculate* the *total value* of a *shipment*"

   or "*check* the *availability of seats* on a *flight*"

3. The granularity of each feature should be such that it would take no more than two weeks to develop; otherwise it will be broken down into smaller features.

FDD: Feature Sets

1. Each feature is identified as a *Step* in one or more *Activities* (also called *Feature Sets*), which are expressed as conforming to the general template:

   <action><-ing> a(n) <object>

   for example: *"reserving a seat"*

2. Activities in turn belong to *Areas* (or *Major Feature Sets*) which are expressed using the general template:

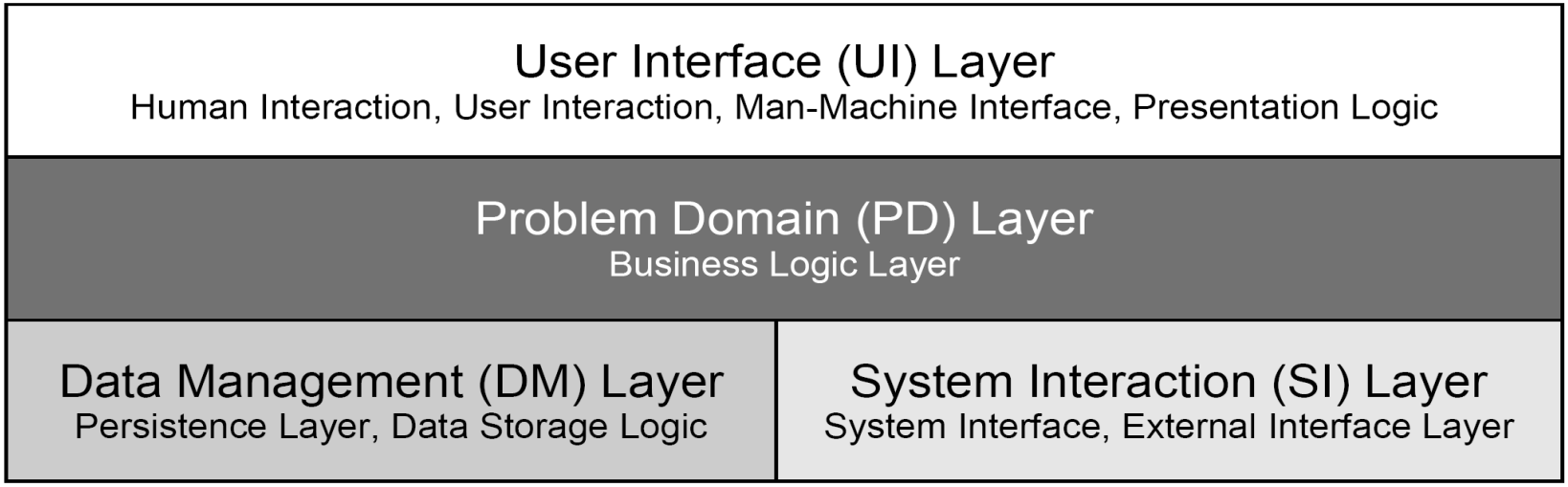
   <object> management

   for example: *"reservations management"*

- The three-layered architecture of Areas-Activities-Steps allows the developers to adequately manage the complexity of the requirements.

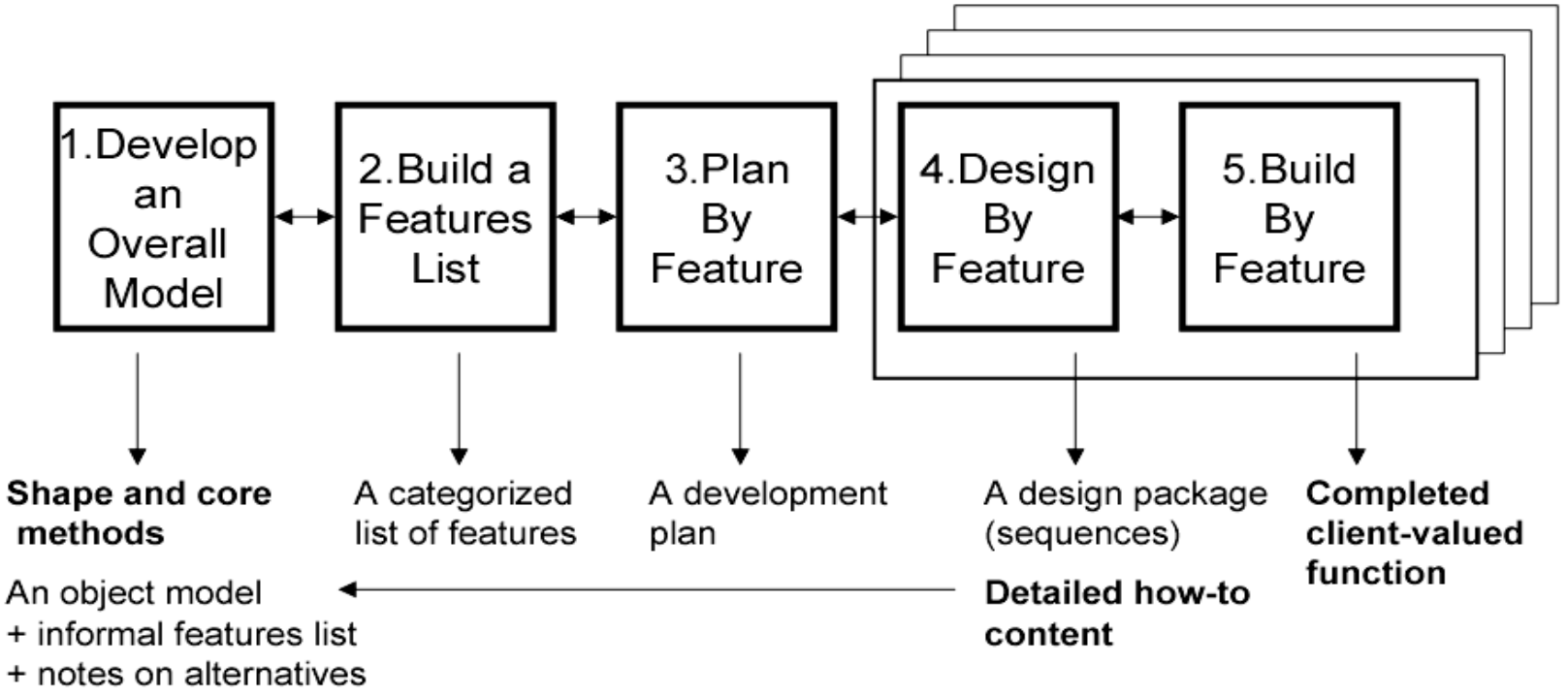
- Features can also be partitioned according to the architectural layer to which they belong.

FDD: Proposed System Architecture



[Palmer and Felsing 2002]

FDD: Process



[Palmer and Felsing 2002]

FDD: Sequential Subprocesses

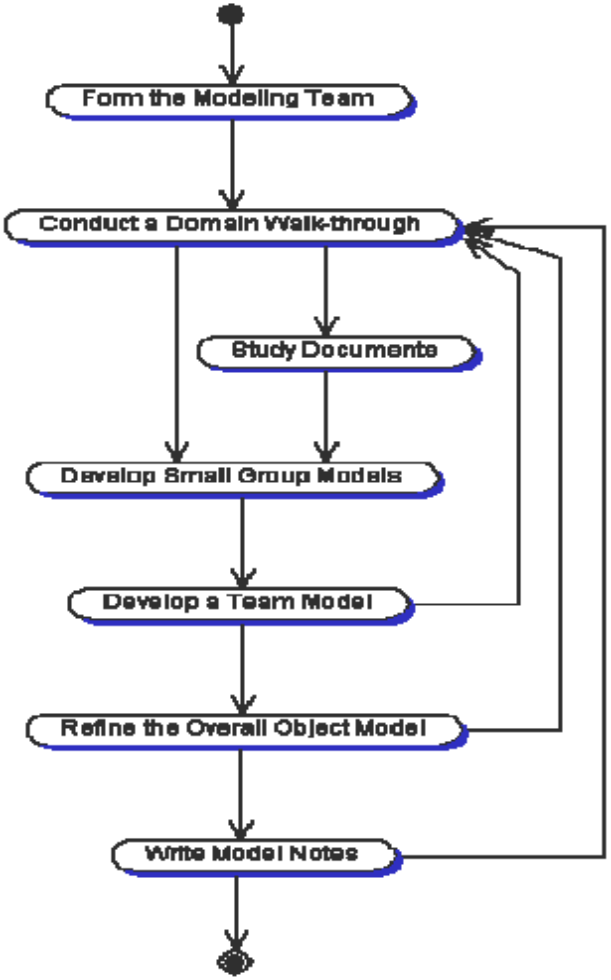
1. ***Develop an Overall Model:*** building a mainly structural model of the problem domain called the Object Model.

2. ***Build a Features List:*** identifying the required functionality of the system, expressed as features and feature sets.

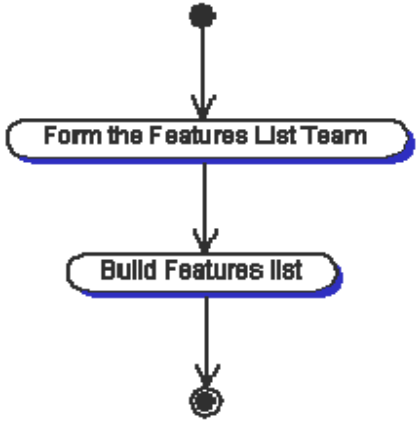
3. ***Plan by Feature:*** scheduling the features for development, and assigning the feature sets (activities), and the classes in the object model, to developers.

FDD Process: Sequential Subprocesses

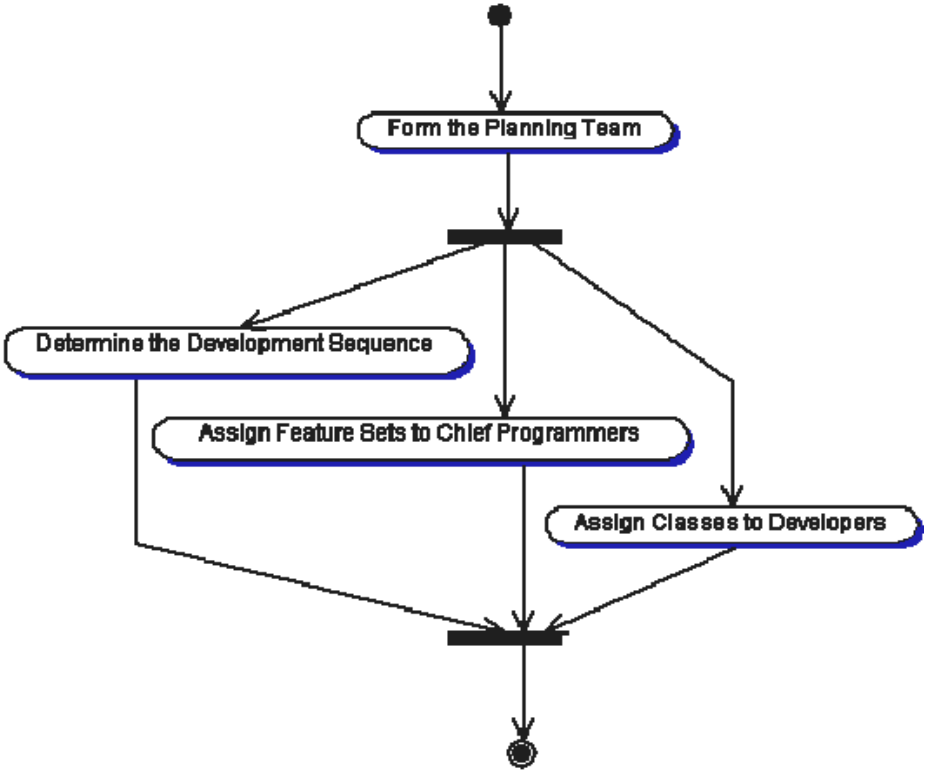
[Develop an Overall Model]



[Build a Features List]



[Plan by Feature]



[Palmer and Felsing 2002]

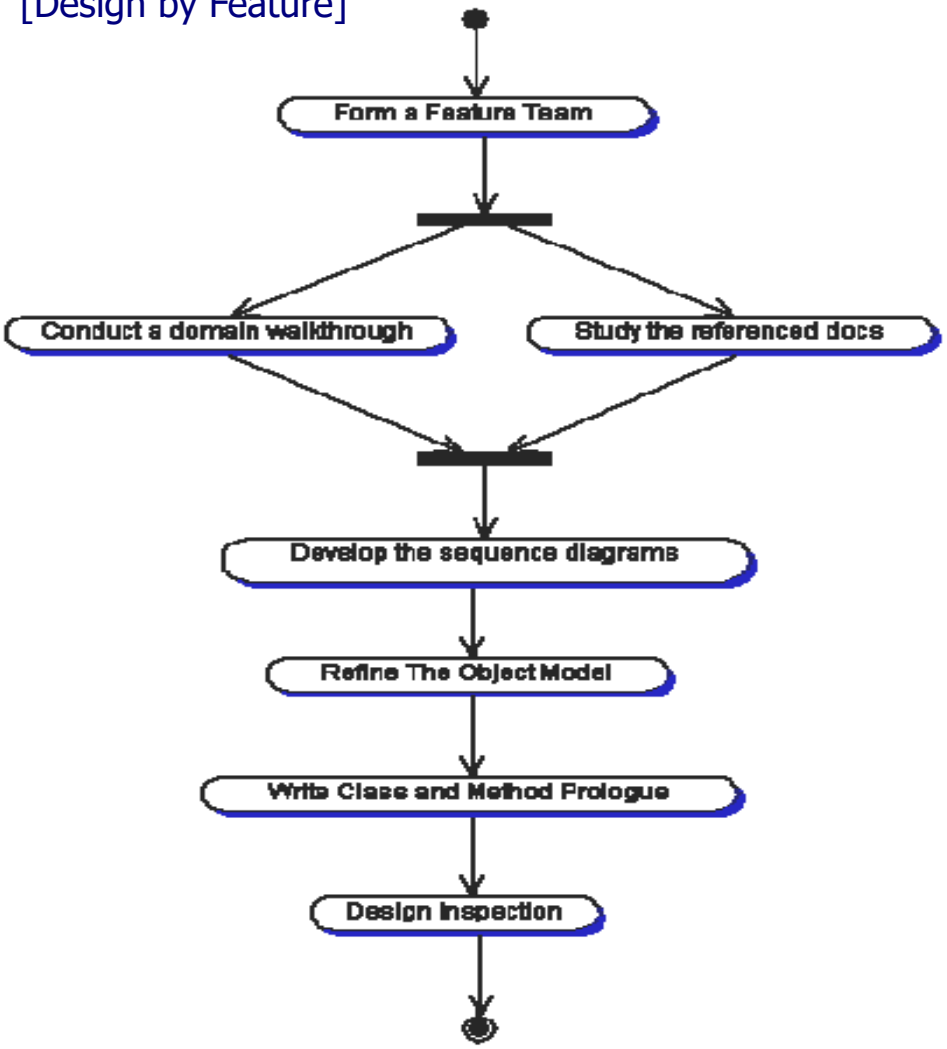
FDD: Iterative Subprocesses

4. ***Design by Feature:*** determining how the features should be realized at run-time by interactions among objects.

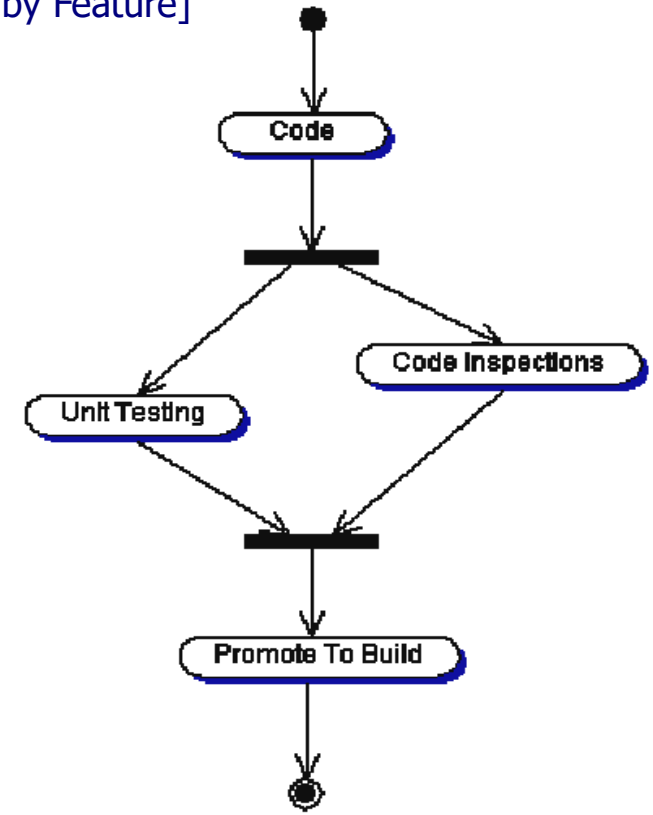
5. ***Build by Feature:*** coding and unit-testing the necessary items for realization of the features. Implemented items are then promoted to the main build.

FDD Process: Iterative Subprocesses

[Design by Feature]

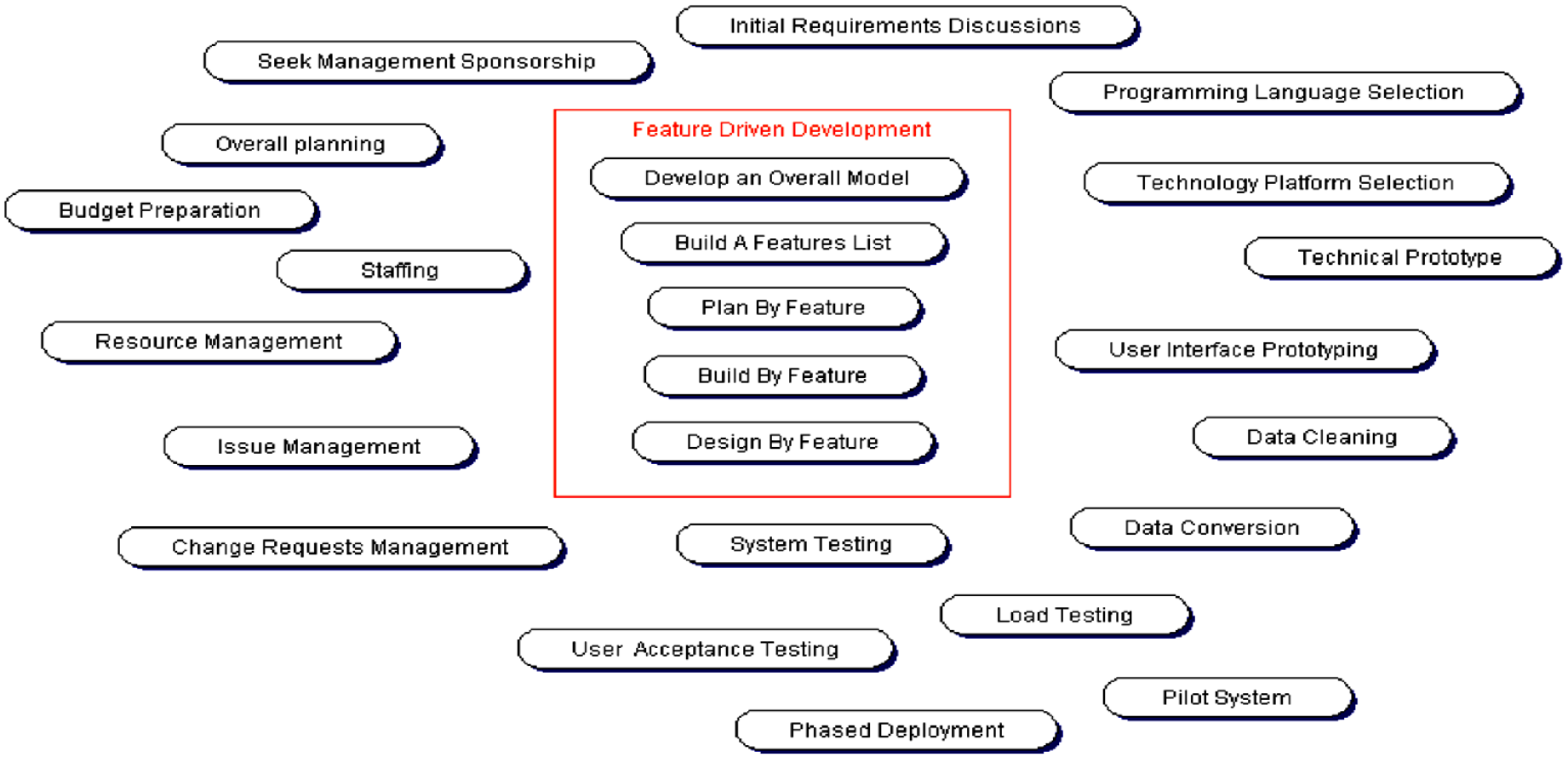


[Build by Feature]



[Palmer and Felsing 2002]

FDD Process: Surrounding Activities



[Palmer and Felsing 2002]

FDD Process: Develop an Overall Model

1. Form the Modeling Team, consisting of several development experts (Chief Programmers), and one or more Domain Experts. The team operates under the guidance of a modeling expert (Chief Architect).
2. Iterate the modeling cycle and produce the Object Model:
    1. An overview of the problem domain is presented by Domain Experts.
    2. The problem domain is partitioned into areas, and the modeling is performed for each area through iterating the following tasks:
        1. Conduct a domain-area walkthrough, aided by Domain Experts.
        2. Study documents of the problem domain area (if available).
        3. Develop small Group Models of the problem domain area.
        4. Develop a Team Model of the area by approving one of the group models or merging ideas from amongst them.
        5. Integrate the model produced into the overall Object Model.
        6. Write model notes, which describe specific aspects of the model that are not explicitly addressed by the model itself.

FDD Process: Build a Features List

1. Form the Features-List Team, which consists of the Project Manager and the Chief Programmers participating in the modeling team.
2. Build the Features List, which is a three-layered hierarchical list with the following structure:
    1. A list of *areas* (major feature sets).
    2. For each area, a list of *activities* (feature sets) within that area.
    3. For each activity, a list of features as the *steps* in the activity.

- The features-list is built in a top-down fashion:
    1. *Areas* (high-level feature sets) are identified by carefully investigating the knowledge acquired about the problem domain, particularly the problem-domain areas (partitions).
    2. *Activities* (low-level feature-sets) in each area, and *steps* (features) in each activity are identified by applying functional decomposition.

FDD Process: Plan by Feature

1. Form the Planning Team, consisting of the Project Manager, Chief Programmers, and a Development Manager, which is put in charge of the development effort and supervises the Chief Programmers.
2. Determine the development sequence by scheduling the development of the feature sets (activities), specifying a date for the completion of each. A completion date is then determined for each area (major feature set).
3. Assign feature sets to Chief Programmers, thereby declaring them as the owners of the feature-sets assigned to them.
4. Assign classes to developers, thereby declaring them as Class Owners.

FDD Process: Iterations

- Feature-set-developers (Chief Programmers) will develop the feature sets assigned to them by commissioning class-developers (called Class Owners) to cooperate in order to design and implement the features.
- Strands of design-and-build iterations start off as each Chief Programmer selects the set of features (called the Work Package) that should be developed in each of the iterations performed under his supervision, and forms a team of Class Owners to do the job.
- A Chief Programmer selects features and schedules his iterations according to the overall development plan, taking care that each iteration takes no longer than two weeks to complete.
- At any point during this development period, several iterations are being performed concurrently, some of them supervised by the same Chief Programmer, with each of the Class Owners taking part in several iteration-teams simultaneously.

FDD Process: Design by Feature

1. Form a Features Team: the Chief Programmer brings together the owners of the classes that might be involved in the realization of the selected features.
2. Conduct a domain walkthrough (if at all necessary).
3. Study the referenced documents (if at all existent).
4. Develop Sequence Diagram(s), which show how objects should interact at run-time in order to implement each of the features.
5. Refine the Object Model (class diagrams) so that it supports the sequence diagrams produced in the previous task.
6. Write Class- and Method-Prologues for the elements of the model.
7. Design inspection is performed by the features team.
8. A Design Package is produced, consisting of: sequence diagrams, refinements made to the object model, prologues, and notes on the design alternatives, constraints, and assumptions.

FDD Process: Build by Feature

1. Implement classes and methods according to the specifications given in the Design Package. Each of the Class Owners implements the necessary items (including the unit-testing code).
2. Conduct a code inspection, either before or after the unit-test, during which the features team examines the code to make sure of its integrity and conformance to coding standards.
3. Unit-test the code to ensure that all classes satisfy the functionality required. Class Owners perform class-level unit-tests, as well as feature-level unit-tests prescribed by the Chief Programmer.
4. Promote to the build, if the implemented classes are successfully inspected and unit-tested.

FDD: Strengths and Weaknesses

- ***Strengths***
  - Iterative-incremental process
  - Based on a general layered architecture for systems
  - Based on structural and behavioural modeling of the problem domain
  - Based on system requirements captured as *Features*
  - Traceability implemented through using *features* as a basis throughout the process
  - Simple and straightforward process, yet well thought-out
  - Only mild model-phobia

FDD: Strengths and Weaknesses

- ***Strengths*** **(Contd. 1)**
  - Seamlessness observed throughout the process via *feature*-based modeling activities
  - Design-based development
  - Continuous validation
  - Frequent deliveries once the iterations start
  - Complexity management at the *features* level through layering

FDD: Strengths and Weaknesses

- ***Strengths*** **(Contd. 2)**
  - Continuous integration
  - Modeling at the problem-domain-, system-, inter-object-, and intra-object levels
  - Group modeling used as a technique for putting all the people involved in the overall picture
  - Iterative modeling in order to enhance the accuracy, completeness and consistency of the models

FDD: Strengths and Weaknesses

- ***Weaknesses***
  - Does not cover post-implementation activities and preliminary analysis.
  - Lacks adaptability due to inexistence of iteration-level planning, reviewing and revision.
  - Intensive project supervision is essential.
  - No formalism

References

- Palmer, S. R., Felsing, J. M., *A Practical Guide to Feature-Driven Development*. Prentice-Hall, 2002.
- Hunt, J., *Agile Software Construction.* Springer, 2006.