2. Incremental Development Paradigm

- There are many situations in which initial software requirements are reasonably well defined, but the overall scope of the development effort precludes a purely linear process.
- In addition, there may be a compelling need to provide a limited set of software functionality to users quickly and then refine and expand on that functionality in later software releases.
- In such cases, you can choose a process model that is designed to produce the software in increments.

There are two types of Incremental development: The Incremental model and The Rapid Application Development (RAD) Model.

2.1 Incremental Model

- The incremental model applies linear sequences in a staggered fashion as calendar time progresses.
- Each linear sequence produces a deliverable "increment" of the software.
- For example, word-processing software developed using the incremental paradigm might deliver:
  - basic file management, editing, and document production functions in the first increment;
  - more sophisticated editing and document production capabilities in the second increment;
  - spelling and grammar checking in the third increment; and
  - advanced page layout capability in the fourth increment.

- When an incremental model is used, the first increment is often a *core product.*
- That is, basic requirements are addressed, but many supplementary features (some known, others unknown) remain undelivered.
- The core product is used by the customer (or undergoes detailed review).
- As a result of use and/or evaluation, a plan is developed for the next increment.
- The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality.
- This process is repeated following the delivery of each increment, until the complete product is produced.

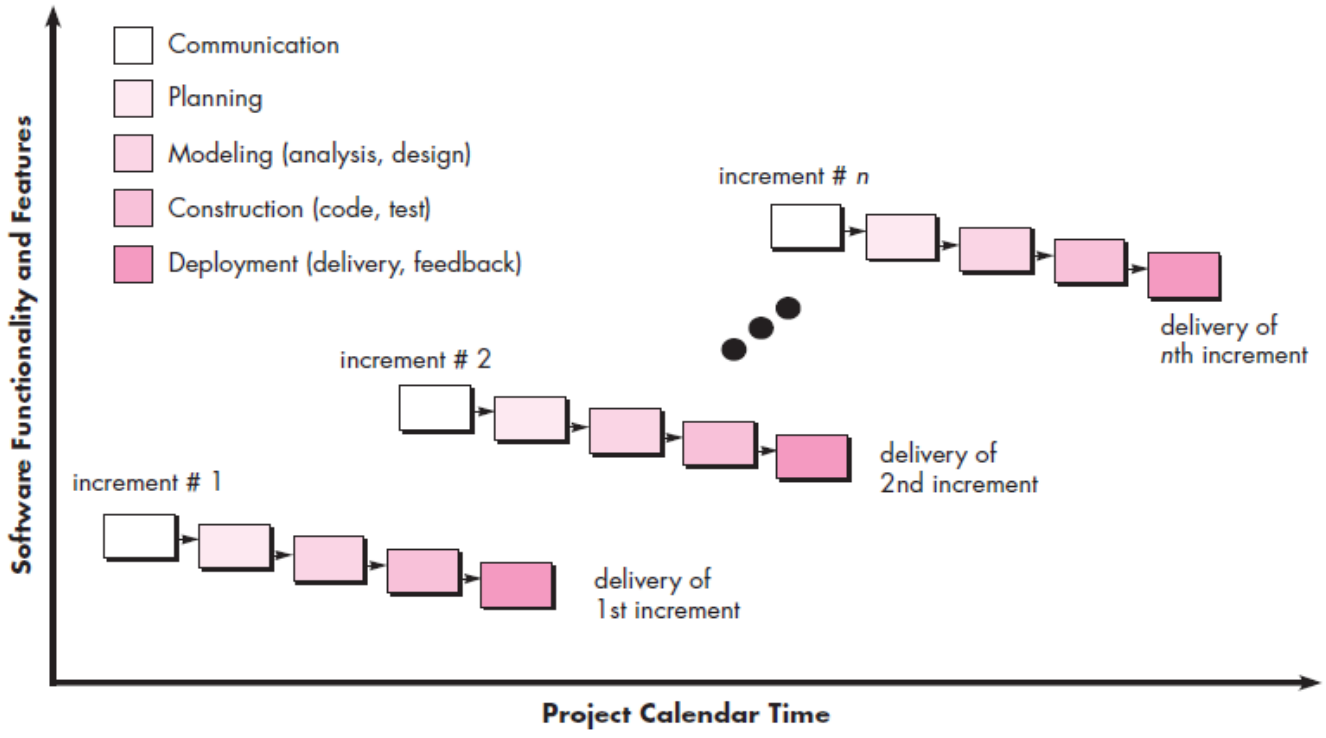

Figure: Incremental Model

- The incremental process model is iterative in nature.
- The incremental model focuses on the delivery of an operational product with each increment.
- Early increments are stripped down versions of the final product, but they do provide capability that serves the user and also provide a platform for evaluation by the user.
- Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project.
- Early increments can be implemented with fewer people.
- If the core product is well received, then additional staff (if required) can be added to implement the next increment.
- In addition, increments can be planned to manage technical risks.
- For example, a major system might require the availability of new hardware that is under development and whose delivery date is uncertain.
- It might be possible to plan early increments in a way that avoids the use of this hardware, thereby enabling partial functionality to be delivered to end-users without inordinate delay.

Advantages of Incremental model:

1. It generates working software quickly and early during the software life cycle.
2. Flexibility is more and less costly.
3. Testing and debugging becomes easier during a smaller iteration.
4. Risk can be managed more easily because they can be identified easily during iteration.
5. Early increments can be implemented with fewer people.
6. After every iteration any faulty piece software can be identified easily as very few changes are done after every iteration.
7. It is easier to test and debug as testing and debugging can be performed after each iteration.
8. This model does not affect anyone's business values because they provide core of the software which customer needs,which will indeed help that person to keep run his business.
9. After establishing an overall architecture, system is developed and delivered in increments.

Disadvantages of Incremental model:

1. Each phase of an iteration is rigid and do not overlap each other.
2. Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.
3. If the requirements initially were thought to be stable but at later stages are realized to be unstable then the increments have to be withdrawn and have to be reworked.
4. Resulting cost may exceed the cost of the organization.