**Submitted By,**
**Adesh Oak**
**(aoak@iu.edu)**

# LECTURE SUMMARY
# WEEK 2

- A generic process model consists of software process which has a particular process framework which performs umbrella activities which enclose a set of framework activities. These framework activities can be further microscopically viewed as a collection of actions and task sets containing work tasks.
- The process flows in process models can be of several types viz.
  1. Linear Process Flow
  2. Iterative Process Flow
  3. Evolutionary Process Flow
  4. Parallel Process Flow

**<u>Prescriptive Process Models: -</u>** Generally plan-driven models that incorporate orderly or structured approach to software development. Following this type of model leads to two legitimate questions –

1. How will this model cope with the software world where things are constantly changing since the backbone of this model is orderly and structural approach?
2. If we ditch this model with a less structured model, how can interconnection and coherence between the components and requirements be achieved in software development?

**<u>Difference between Incremental and Iterative Approach -></u>**

Incremental – Design, Implementation and testing carried out through small incremental releases.
e.g., while writing an essay, one might write it perfectly by writing a perfect line every time.

Iterative – Iteratively work and improve the product.
e.g., while writing an essay, we might want to write a rough draft first to get the idea and then work on improving it through readings and analysis phases.

*Mona Lisa painting was discussed as an example for this in class*

There are few major types of Prescriptive Models: -

1. Waterfall Model: The Waterfall model is a linear, sequential software development method in which development can be perceived as flowing steadily downwards through the phases of requirements, design, implementation, testing, deployment and maintenance (just like in a waterfall water flows from top to down). This model is best suited for projects with well-defined and fixed requirements, where the scope and requirements of the project are clearly understood before development begins.

2. Prototyping Process Model: The Prototyping Model is a software development process in which a prototype, or working model, of a product is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved. This model is best suited for projects with unclear or changing requirements, where the scope of the project is not well understood, and the customer's needs are not clear until the final product is delivered.

3. Spiral Process Model: The Spiral model is a software development process that combines elements of both the Waterfall model and the Prototyping model. It is a risk-driven process model that emphasizes on the iterations of development and allows for the concurrent consideration of technical and business risks. This model is best suited for projects that require a high level of risk management and are complex in nature.
   It also has its own set of issues that can arise during implementation. These include: complexity, high resource requirement, high cost, time-consuming, risk management not being effective, difficulty in communication and coordination, lack of flexibility, scalability issues, not suitable for smaller projects or projects with a limited budget and timeline, and not being the best choice for projects with well-defined scope and requirements.

4. Unified Process Model: The Unified Process (UP) is an iterative and incremental software development process that emphasizes a collaborative and adaptive approach to software development. It is based on the object-oriented design and is a widely used methodology for software development. It is divided into four phases: Inception, Elaboration, Construction and Transition.
   1. Inception: Defining the scope and objectives of the project.
   2. Elaboration: Identifying and mitigating risks and defining the architecture of the project.
   3. Construction: Building and testing the software components.
   4. Transition: Deploying the software and providing post-deployment support.

   These phases are further divided into smaller iterations called time boxes, which allows the project team to adapt to changing requirements and risk.

Following this, pros and cons of each model were studied:

1. Waterfall Model:
   Pros
   - ➔ Easy to understand and use
   - ➔ Suitable for projects with well-defined and fixed requirements

   Cons
   - ➔ No room for change, lack of flexibility
   - ➔ No feedback from the customer
2. Prototyping Model:
   Pros
   - ➔ Suitable for projects with unclear or changing requirements
   - ➔ Provides customer feedback throughout the development process

   Cons
   - ➔ Can be time-consuming, costly
   - ➔ Can lead to scope creep
   - ➔ Can lead to incomplete or inadequate final product and temptation to “ship” prototype
3. Spiral Model:
   Pros
   - ➔ Incorporates risk management
   - ➔ Suitable for complex projects

   Cons
   - ➔ Project hard to manage
   - ➔ Requires expert team, costly
4. Unified Process Model:
   Pros
   - ➔ Collaborative and adaptive approach
   - ➔ Object-oriented design, with emphasis on documentation

   Cons
   - ➔ High resource requirement, high overheads
   - ➔ Overlapping phases may cause problems
   - ➔ Use cases not precise

There is a further, more optimized type of Unified Process Modeling called as:
**<u>Rational Unified Process Modeling (RUP):</u>**

The Rational Unified Process (RUP) is a software development methodology that provides a framework for organizing, planning, and controlling the process of software development. It is based on the idea of iterative and incremental development, where the project is divided into multiple phases, each of which is focused on a specific aspect of the project. Each phase produces a set of software artifacts, which are the tangible outputs of the phase. These artifacts include things like requirements documents, design documents, and code. The goal of RUP is to provide a structured, repeatable process for software development that helps to ensure that the final product meets the needs of the customer and is delivered on time and within budget.

Some software artifacts in RUP during various stages include: -

1. Inception – Vision, Business Case, Risk List, Software plan, Iteration plan, Main used cases
2. Elaboration – Prototypes, risk list, development plan, tools, test suite etc...

Further elapsed time in RUP is the total time taken by project (includes work hours as well as weekends) and effort is the actual amount of work time of the project. It varies for each phase.

## <u>Agile Software Development:</u>

Agile software development is a methodology that emphasizes flexibility, collaboration, and rapid iteration in the software development process. It originated in the late 1990s as an alternative to the traditional, heavily planned and stage-driven approaches to software development such as the Waterfall model. The Agile Manifesto, a set of guiding values and principles for Agile development, was published in 2001. Agile methodologies, such as Scrum and Kanban, have since become widely adopted in software development as they allow for a more flexible and responsive approach to changing requirements, and encourage continuous improvement and team collaboration. Agile methodologies have been proven to be effective in delivering software products in an efficient and timely manner and have become the industry standard for software development.

The **cost** for incorporating changes after the breakpoint in agile method is comparatively low and doesn't increase as rapidly or exponentially as traditional software.

<u>Agile Software Process:</u>

- The Agile software process is a methodology that emphasizes flexibility, collaboration, and rapid iteration in the software development process.
- It is based on the Agile Manifesto, which outlines four core values: individuals and interactions, working software, customer collaboration, and response to change.

- Agile processes are iterative and incremental, with a focus on delivering small, usable portions of the software (referred to as "sprints" in Scrum) on a regular basis.
- This approach allows for a more flexible and responsive approach to changing requirements and encourages continuous improvement and team collaboration.
- Agile methodologies also place a strong emphasis on regular meetings and communication, such as daily stand-ups, retrospectives, and planning meetings.
- The goal of the Agile process is to deliver a working software product that meets the needs of the customer in an efficient and timely manner.

Agility Principles:

Agility in software development refers to the ability to adapt to changing requirements and to deliver working software quickly and efficiently. The core principles of agility include:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

**Tao Te Jing (Daodejing):**

Tao Te Ching, a Chinese text written by Lao Tzu in the 6th century BCE, can be seen as a guide for embracing change and fostering a sense of flexibility, adaptability, and balance in the software development process. It aligns with Agile principles of responding to change, simplicity and continuous improvement. It promotes the idea of embracing the unknown and taking an adaptive approach. It's not directly related to Agile software development, but it could be seen as a guide for an agile mindset.

**Politics of Agile:**

The politics of Agile software development refers to the various power dynamics and decision-making processes that can occur within an Agile team. Agile methodologies promote collaboration, transparency, and inclusivity, but in practice, there can be tensions and conflicts between team members, stakeholders, and management. These politics can manifest in issues such as scope creep, competing priorities, and resistance to change. To mitigate these issues, it is important to have clear roles and responsibilities, open communication, and a culture of continuous improvement. Additionally, it is important to ensure that all team members have a voice and that decisions are made through a consensus-based process. It's important to have a shared understanding of the goals of the project and the principles of Agile development, and to make sure that everyone is aware of their responsibilities and how they contribute to the project.

**<u>Scrum:</u>**
Scrum is an Agile methodology for managing and completing complex projects. It is a framework that provides a structure for organizing, planning, and controlling the process of software development. Scrum is based on the principles of transparency, inspection, and adaptation. It provides a set of roles, events, and artifacts that help to ensure that the project is delivered on time and within budget.

<u>Scrum framework:</u>
The Scrum framework is composed of three roles: the Product Owner, the Scrum Master, and the Development Team. The Product Owner is responsible for defining and prioritizing the features of the product, the Scrum Master is responsible for facilitating the process and removing any obstacles that the team may encounter, and the Development Team is responsible for delivering the product.

<u>Scrum Events:</u>
Scrum includes several events, such as Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective, that help to ensure that the project is progressing as planned and that any issues are identified and addressed in a timely manner. The goal of the Scrum process is to deliver a working product increment at the end of each sprint, which is typically two to four weeks.

In short, Scrum is an Agile methodology that provides a framework for organizing, planning, and controlling the process of software development. It emphasizes transparency, inspection, and adaptation, and helps to ensure that the project is delivered on time and within budget.

---