Assignment -02

1. Pick two of these models: Component-based, Unified Process, or Agile. Using the web, for each model you picked find a real-world example software project that was implemented using the model. For each project:

a. provide a summary of the project

b. explain why the chosen process model was appropriate by highlighting the benefits of the chosen process model and its impact on the project

c. discuss the impact to the project if a different process model (any we discussed in class) had been used with the project.

Component Based Model – real world example – Microsoft's COM:

A. Summary of the project:

It is a platform-independent object-oriented system, used for creating binary software components. These components are created in such a way that they can be interacted with each other. It is not an object-oriented language; it is a standard. This specifies an object model and programming requirements that can be used to create component objects.

The COM objects can be created with a variety of object-oriented programming languages like C++. The created objects can be within a single process, multiple processes, or remote systems. COM is the foundation for many Microsoft based frameworks such as Windows Shell.

B. **Why the chosen process model was appropriate:**

The Component-based model is appropriate for this project as it allowed the development team to divide the complex system into smaller components. It also allowed the team to maintain and manage the components. This not only makes the team develop all the individual components separately, but also reduces the risk and complexity of the process involved.

One more advantage of using this model for this project is that it allows re-using the preexisting software instead of requiring the team to create the same functionality from the scratch multiple times. For example, using COM model COM+, an extension of COM has been developed. COM allows the reuse of objects without the knowledge of their implementation. Component based model has five major characteristics – Reusability, replaceability, extensibility, encapsulation, and independence. Microsoft's COM takes advantage of these features and gives an essence of language-neutral manner of implementing objects which can be used in a variety of environments, even that are different from one another allowing the cross communication.

C. The impact to the project if a different process model had been used with the project:

For this project, the development team would have emphasized on delivering smaller and incremental versions of the product by worrying about delivering the project in less time if an alternative process model, such the Agile model, had been employed. Although this might have assisted in adapting to shifting client needs, it might not have been suitable for a complicated system like the Microsoft's COM. This would have required the team to focus on providing smaller components first and then integrating them. The Agile methodology would have led to a lengthier development cycle, increasing development costs, and lowering user satisfaction, which is the exact contrary of its implementation in software engineering. Likewise, the project would have taken longer to build if the Unified Process—a more organized and staged method to software development—had been utilized. Because of this, COM might have been published later than other products, losing market share. The Unified Process also emphasizes detailed documentation, which might have slowed down software development and made it more challenging to incorporate modifications.

Agile Based Model – real world example – Apple's MacBook

A. Summary of the project:

The MacBook is a line of laptops designed and marketed by Apple Inc. The first MacBook was introduced in 2006 and has since become one of the most popular notebook computers in the world. The current lineup consists of MacBook Air and MacBook Pro with several specifications, features, and upgrades from previously released ones.

B. Why the chosen model was appropriate:

The Agile process flow allows for a more flexible and adaptive development approach, making it suitable for the creation of the MacBook. A more effective and user-centered development process was made possible by the Agile approach, which places a strong emphasis on cooperation between developers, designers, and product owners. The Mac's user experience was enhanced, and the device's competitiveness was preserved. This is because of the Agile model's rapid updates and enhancements based on user feedback. The use of the Agile process paradigm contributed significantly to the MacBook's success by enabling a quicker and more effective development process and enhancing the user experience as a whole.

C. The impact to the project if a different process model had been used with the project:

The Mac may have taken longer to build if an alternative process model, such as the component-based model, had been employed. This is because the component-based approach stresses the creation of individual components, which could have slowed down the development process. Additionally, the development process may not have been as flexible or adaptable using the component-based methodology, which would have led to a less user-focused end product.

References:

1. https://cs.ccsu.edu/~stan/classes/CS530/Notes18/16-CBSE.html
2. https://en.wikipedia.org/wiki/Component_Object_Model
3. https://www.mendix.com/blog/what-is-component-based-architecture/#:~:text=The%20benefits%20are%20reduced%20development,replacing%20components%20without%20major%20disruption.
4. https://tryqa.com/what-is-agile-methodology-examples-when-to-use-it-advantages-and-disadvantages/
5. https://digitalcommons.njit.edu/cgi/viewcontent.cgi?article=1471&context=dissertations

2. Working remotely has affected software development especially agile development. Review these links (HERE, HERE) and find additional resources for your research and share your observations on 'How remote environments have affected Agile Development.' Also discuss a list of best practices to make agile development more effective for remote environments.

The article: https://www.netguru.com/blog/how-to-deal-with-agile-scrum-in-a-remote-environment, debunks the myth that Scrum and Agile forbids from practicing remote work environments. Co-location is recommended because a face-to-face conversation is an efficient and effective method of exchanging information among the team members as well as the project stakeholders. However, it is not entirely possible to have the stakeholders and team working at the same location. That alongside Covid19 had led us to choose the option of working, connecting, and collaborating with everyone remotely. As per Agile Manifesto Principles also, the development team is required to have a face-to-face conversation.

To run and maintain remote agile, a lot of features and techniques must be abstracted from the physical agile environment. This makes it harder to manage the same features in remote ambience. It makes the team less self-aware and less mature. However, the agile definition itself subtly promotes the idea of remote organization in a way that agile must be able to adapt to drastically changing market. And, in our case Covid19 has swiftly changed the manner an organization works. The organizations had to quickly switch to the remote environment and continue to have hassle free work delivery. Simply put, co-located work is an aggregation of many like-minds working under one roof.

Having a close-knit group working in the same place enables instant communication which in turn builds quick trust, simplifies problem solving and allows fast-paced decision making. As per the Harvard Business Review: Workplace Trends; Zoltan Lippenyl and Tanja van der Lippe, the experience of remotely working environment had led to inefficiency in working as a whole. As per the survey in the article "Co-workers working from home and individual and team performance" New Technology, Work and Employment March 2020, it is stated that 80% of the survey takers have responded that they would have better relationships if it was for frequent team communications. About 84% respondents stated that working remotely had delayed the challenges and concerns they were facing. Around 41% people said bad things about their co-workers when working remotely as compared to only 31% of co-located employees. This necessarily points out that working in the same location would be much more recommended for strong professional relationships and it would avoid communication gap among the team.

The abrupt shift from co-located workspace to remote environment that was brought by coronavirus majorly affecting how the agile team is managed. Other negative impact of working remotely is the burnout. About 82% of remote workers in the U.S. experience work fatigue by reporting only half the work assigned to them but working for longer hours. This has ultimately affected not only the work efficiency, but also the well-being of the employees. Some other challenges include stress induced health problems and other technical issues with the worker's remote system setup. The shift to remote behavior also blurred out the work-life balance of employees by their superiors ordering them to report the work even at odd times in many start-ups, that would have run in a smooth fashion if it wasn't for Covid-19 and remote work.

Despite these difficulties, numerous Agile development teams have adapted to the new remote environment and are still producing high-caliber work. Agile development can flourish in a remote context by utilizing remote collaboration tools, putting into practice agile methods and practices that are easily adaptable to a remote environment, and keeping an emphasis on transparent communication and responsibility. Stronger connections among team members can be built by having more space to get to know each other. The weekly calls can be made a little longer to talk about vacations, weekend plans or share what they did over the weekend. Alongside this, a team outing can impact positively on the way the team works.

Best practices to make agile work in a remote environment:

1. Review the situation: Evolving the team culture by arranging a videoconference once in a while.
2. Engage as a team: Be attentive, in the present and involve in the team engagement activities.
3. Foster transparency: Be understanding, empathetic and communicate openly as and when needed.
4. Leverage technology: Understand and upskill the remote work technology.
5. Evolve team practices: Keep in touch with the remote team members, schedule one-on-one meetings and check-in on them.
6. Maintain momentum and enthusiasm: Instead of forgetting the reason to do work, try to solve end-user problems. Try to remove the feeling of remote environment acting as a barrier between the team and end-users.

References:

1. https://www.lucidchart.com/blog/disadvantages-of-working-from-home#:~:text=For%20full%2Dtime%20remote%20employees,than%20those%20in%20the%20office.
2. https://www.forbes.com/sites/forbesbusinesscouncil/2022/08/31/dont-forget-the-downsides-of-remote-work/#:~:text=Being%20unable%20to%20disconnect%20can,office%20workers%20reported%20the%20same.
3. https://www.jobstreet.com.my/en/cms/employer/laws-of-attraction/inspirations/addressing-common-work-from-home-issues-to-keep-employees-

motivated/#:~:text=Blurred%20work%2Dlife%20balance&text=Employees%20are%20often%20expected%20to,work%20can%20also%20be%20disorienting.
4. https://www.gartner.com/smarterwithgartner/6-best-practices-for-remote-work-by-agile-software-development-teams

3. Read the following white paper "The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments" and summarize your observations. Then review a real-world project and explain how the lessons from the above paper were (or can be) adapted to the project. Provide your thoughts (what worked or did not work and possibly more suggestions.)

The article emphasizes the significance of extending agile methodologies to encompass the whole software delivery process and illustrates how the Scrum approach can be expanded in this regard. In actuality, the primary obstacle that organizations encounter when scaling agile development is the extent of the delivery cycle. The article then explains the eight factors involved in scaling agile software delivery and their impact on adapting agile methods effectively to meet the unique needs of the organization. The growing popularity of agile development is due to its proven results - organizations have found that compared to traditional project teams, agile teams boast higher success rates, produce higher-quality work, receive greater satisfaction from stakeholders, generate a better return on investment, and bring their systems to market faster. Agile approaches are now being used in a multitude of scenarios, beyond the small, co-located team environments that were primarily discussed in early agile literature. Rather, agile techniques are being implemented across the entire software delivery process.

A group of 17 methodologists created a manifesto and a set of supportive principles aimed at promoting improved methods of software development. The manifesto outlines four values and twelve principles that serve as the basis of the agile approach. The Agile Manifesto states that the authors have found more effective ways of developing software through their own experiences and by assisting others. They have come to value the following:

1. People and communication over processes and tools.
2. Producing working software rather than comprehensive documentation.
3. Engaging in collaborative customer interaction instead of negotiating contracts.
4. Being flexible and adapting to change instead of sticking to a strict plan.

The values outlined in the Agile Manifesto are backed by a set of 12 principles. Agile software development is an iterative and incremental approach that consistently delivers high-quality software in a cost-effective and efficient manner, using a process that is driven by delivering value. A prevalent issue in many organizations is that informal, ad-hoc teams often claim to be using agile methods, based on having read a couple of articles about agile development. They misinterpret agility as any form of freestyle, informal software creation without proper documentation. These ad-hoc teams often encounter problems, which can reflect poorly on legitimate agile teams.

The Agile Scaling Model is a flexible framework for successful implementation and customization of agile practices for a system delivery team of any size. The Agile Scaling Model (ASM)

categorizes different levels of agile software development into three groups: Core Agile Development, Disciplined Agile Delivery, and Agility at Scale.

1. Core Agile Development involves the use of basic agile methods such as Scrum or Agile Modeling, which are best suited for small, co-located teams working on simple systems. These methods have a value-driven system development lifecycle (SDLC) and self-governing practices like daily stand-up meetings and requirements envisioning.
2. Disciplined Agile Delivery covers the entire software development process, from project inception to transitioning the system into production or the marketplace. This category includes processes like Dynamic System Development Method (DSDM) and Open Unified Process (OpenUP) and focuses on small, co-located teams working on straightforward systems. To address the full delivery lifecycle, this category requires a combination of core agile methods and tailored traditional practices.
3. Agility at Scale takes Disciplined Agile Delivery to the next level by addressing the challenges posed by scaling factors such as team size, geographical distribution, regulatory compliance, organizational complexity, technical complexity, and more. This category involves tailoring disciplined agile delivery practices and adopting new practices to address the additional risks of working at scale.

The process of scaling agile involves transitioning from limited methods to a comprehensive and structured disciplined agile delivery approach. The next step involves determining the level of complexity involved. Having an efficient software building process is great, but it won't matter if there's a lack of agreement on what needs to be built or if it doesn't work with existing infrastructure. We should go beyond just focusing on agile software development and take into account the overall complexity of delivering a solution. A disciplined agile delivery process, which evolves over time, consistently creates high-quality solutions in a cost-effective and timely manner while considering both risk and value throughout the development cycle. Project teams must operate within the rules and regulations established by their organization. An effective governance program should encourage compliance with these guidelines.

Organizations may choose to create their own disciplined agile delivery process by merging elements from Scrum, practices from Extreme Programming (XP), as well as practices from other processes such as Agile Modeling (AM), Agile Data (AD), and Feature Driven Development (FDD). This approach can be effective, but it may also require a significant amount of time and resources. Not every project team is faced with the same scaling factors, nor do they experience the same degree of complexity with these factors. However, these challenges contribute to the overall complexity of a project, and the team must find ways to overcome these challenges. To handle the various complexities in your development organization, it is necessary to adjust your disciplined agile delivery process.

The terms "scaling factors" and "complexity factors" are used to describe different aspects of a project team's challenges in adopting agile strategies. However, when viewed from the perspective of implementing agile strategies across an organization, these factors could both be considered scaling factors. To simplify, the author has chosen to use only the term "scaling factor" to refer to

both types of challenges. This approach aligns with the agile principle of favoring simplicity over perfection. As necessary, adopt updated techniques. Teams that are big in size or spread across different locations can arrange themselves as groups of teams focused on specific features or groups of teams working on specific components. Teams operating in regulatory environments might have to implement practices related to formal documentation.

The priority for teams should be to deliver value, not just to gain personal experience with the latest tools. Agile teams should concentrate on consistently producing desirable outcomes, such as delivering high-quality software that meets stakeholders' requirements efficiently and cost-effectively. The Agile Scaling Model offers guidance for recognizing the challenges encountered when adapting and customizing agile approaches.

In a real-world project, the lessons from the ASM can be adapted by first identifying the project's scaling category, then tailoring the disciplined agile delivery practices to meet the needs of the project. The project team should also regularly review and assess their progress, making adjustments as necessary. In order for the ASM to be successful, it is important to have strong communication and collaboration among the project team and stakeholders. The project team should work together to ensure that everyone understands the goals and priorities of the project, and that the team is aligned on the best approach to achieve those goals. The team should also be willing to be flexible and make changes as needed to ensure the project's success. In conclusion, the Agile Scaling Model provides a useful framework for scaling agile practices in a real-world project. However, the success of the ASM ultimately depends on the project team's ability to effectively apply the framework and make necessary adjustments throughout the project lifecycle.

References:

1. https://www.agilealliance.org/wp-content/uploads/2016/01/Agile-Scaling-Model.pdf
2. https://www.atlassian.com/agile/agile-at-scale
3. https://www.infoq.com/news/2015/11/real-life-scalingBKK/