MONASH University

FIT9138 IS Analysis, Design and Systems Thinking
Week 2 System Development Methodologies (Part 2)

AIMS OF THE APPLIED SESSION

1) Compare and contrast the system development methodologies
2) Discuss the impact of design practice on individuals, communities, organisations, and society

STRUCTURE OF THIS WEEK'S APPLIED SESSION

- Section A: Group Creation Reminder (10 min)
- Section B: Warm-up session (Matching models used in SDLC with the names and definition; Models Reflection) (30 min)
- Section C: Discussion (Compare and contrast: Predictive vs Adaptive SDLC) (40 min)
- **Break (10 min)**
- Section D: Discussion (Scrum vs. XP Scenarios - Choosing the Right Methodology) (20 min)
- Section E: Mini Case (Going with the Flow: Agile Development at Dell) (60 min)

YOUR TASKS

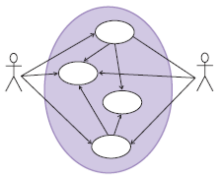
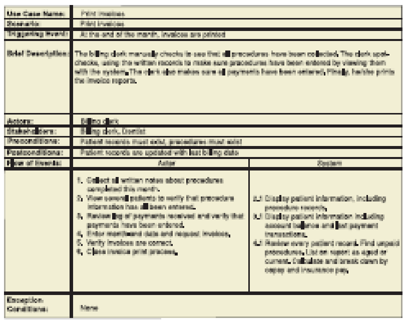
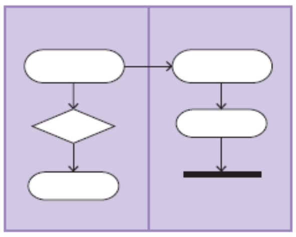
BEFORE YOUR APPLIED SESSION

- Attend/listen to seminar 2
- Read the discussion, mini case, and further reading

IN YOUR APPLIED SESSION

You will work with your fellow students and tutor through the Week 2 Applied activities.

Section A: Group Creation Reminder	Your tutor will remind you to form groups of 5 for the Group Assignment. The groups must be finalized by Week 3.
Section B: Warm-up session (Matching models used in SDLC with the names and definition; Models Reflection)	A model is a representation or abstraction of some aspect of the system being built. There are dozens of different models that an analyst or designer might develop and use. **Activity: Matching models used in SDLC with the names and definition** ● Choose Model Name and Definition that matches the model's picture depicted in the first column ● After you finish, Tutor will guide you to discuss the answers together

Model	Model Name	Model Definition
B1) 		
B2) 		
B3) 		

Use Case Name:	Print invoices	
Scenario:	Print invoices	
Triggering Event:	At the end of the month, invoices are printed	
Brief Description:	The billing clerk manually checks to see that all procedures have been collected. The clerk spot-checks, using the written records to make sure procedures have been entered by viewing them with the system. The clerk also makes sure all payments have been entered. Finally, he/she prints the invoice reports.	
Actors:	Billing clerk	
Stakeholders:	Billing clerk, Dentist	
Preconditions:	Patient records must exist, procedures must exist	
Postconditions:	Patient records are updated with last billing date	
Flow of Events:	Actor	System
	1. Collect all written notes about procedures completed this month. 2. View several patients to verify that procedure information has all been entered. 3. Review log of payments received and verify that payments have been entered. 4. Enter month-end date and request invoices. 5. Verify invoices are correct. 6. Close invoice print process.	2.1 Display patient information, including procedure records. 3.1 Display patient information including account balance and last payment transactions. 4.1 Review every patient record. Find unpaid procedures. List on report as aged or current. Calculate and break down by copay and insurance pay.
Exception Conditions:	None	

Model Name Choices:

1) Activity Diagram	2) Use Case Description	3) Use Case Diagram

Model Definition Choices:

1) describes the various user (or system) activities, the person or component that completes each activity, and the sequential flow of these activities.
2) activity the system performs in response to a request by a user.
3) a textual model that lists and describes the processing details for a use case

Class Discussion (Models Reflection)

B4) Why are graphical models preferred over purely textual descriptions in SDLC?

B5) How can different models (e.g., use case diagrams, use case description, activity Diagram, etc) work together in SDLC?

Section C: Discussion (Compare and contrast: Predictive vs Adaptive SDLC)

Activity: Compare and contrast: Predictive vs Adaptive SDLC

Instructions:

- Read the provided passages carefully.
- Answer the accompanying questions based on your understanding of the passages.
- Be prepared to share and discuss your findings with the class

*Two software development teams, Team Predictive and Team Adaptive, were tasked with building two very different projects***.**

Team Predictive: The Architects

Team Predictive had a clear and structured mission: to create a payroll system for a government agency. The project began with a deep dive into system requirements, where the team collaborated with stakeholders to define what the system needed to achieve. This was followed by the software requirements phase, in which they translated these goals into specific technical requirements. Since the needs were well-understood and unlikely to change, the team moved confidently into analysis, where they identified how the software would fit within the organization's existing processes.

Next came the design phase, where every component was meticulously planned, from user interfaces to database structures. The coding stage was straightforward because developers knew exactly what they needed to build. Once the software was developed, it entered the testing phase, where the team ensured it met all the predefined requirements. Finally, the project culminated in operation, where the software was deployed and handed over for use.

The project's low technical risk stemmed from several factors. First, the technology stack was well-established, with no experimental tools or methods involved. Second, the requirements were so well-defined upfront that the team could avoid major changes during development. Lastly, the project's stable nature reduced uncertainty, allowing the team to predict outcomes with confidence. By carefully planning and executing each phase without overlap, the team minimized surprises and ensured smooth progress.

Team Adaptive: The Pioneers

Team Adaptive faced an exciting yet unpredictable challenge: developing a groundbreaking social media app. Their client envisioned an app that could transform how people connect, share, and communicate, but the specifics of the app were unclear. Early concepts suggested features like real-time video sharing, disappearing messages, and AI-driven content recommendations, but these ideas were expected to evolve as the app took shape.
Because the requirements were uncertain and highly dynamic, Team Adaptive embraced flexibility. They broke the project into small, iterative sprints. Each sprint focused on developing and delivering specific features, such as a "stories" feature similar to Instagram, a real-time chat system like WhatsApp, or a personalized newsfeed algorithm akin to TikTok. At the end of each sprint, the team presented their progress to the client and gathered feedback to shape the next iteration.

This iterative process allowed them to adapt quickly to changing needs and user demands. For instance, when beta testers expressed a strong desire for a collaborative video creation tool, the team was able to pivot and prioritize that feature in subsequent sprints. Though the project involved high technical risks, such as experimenting with cutting-edge AI and cloud-based scalability, their flexible approach ensured they could tackle challenges as they arose.

By embracing uncertainty and responding swiftly to feedback, Team Adaptive ensured the final product was not only innovative but also highly relevant to its users, paving the way for a potentially industry-changing app.

Answer this question by discussing it with your group

In predictive SDLC,

C1) Why are the software requirements/needs typically well-defined and unlikely to change? What factors contribute to the stability of the needs?

C2) Why is planning a critical part of predictive SDLC, ensuring that no subsequent stage can begin before completing the previous one?

C3) What kind of projects are best suited for development using predictive SDLC? Give example.

In adaptive SDLC,

C4) Why are software requirements often uncertain and dynamic? What factors contribute to this uncertainty?

C5) How does adaptive SDLC accommodate the dynamics, changing needs, and user requests within a project?

C6) What kind of projects are best suited for development using adaptive SDLC? Give example.

Section D: Discussion (Scrum vs. XP Scenarios - Choosing the Right Methodology)

Activity: Scrum vs. XP Scenarios - Choosing the Right Methodology

- Please read Appendix 1 and the scenarios below
    - Scenario 1: E-commerce Website for Seasonal Sales

      A retail company wants to launch an e-commerce website to maximize sales during the upcoming holiday season. The website needs essential features like product listings, a shopping cart, payment integration, and user account management. While most requirements are clear, the marketing team may request additional promotional features, such as flash sales and discount codes, closer to the launch.
    - Scenario 2: Mobile Game App

      A startup is developing a mobile game app with unique gameplay mechanics. The game concept is evolving, and the team plans to gather feedback from early testers to refine features. The app may require frequent changes to game mechanics, user interface, and storylines based on user preferences.
    - Choose the right methodology and give your justification

Scenario	Methodology (Scrum/XP)	Justification
E-commerce Website for Seasonal Sales	…	…
Mobile Game App	…	…

Section E: Mini Case (Going with the Flow: Agile Development at Dell)

Activity: Going with the Flow: Agile Development at Dell

source: Dennehy, D., Conboy, K., Gogan J. Going with the flow:Agile development at Dell. *Harvard Business Case*.

Part 1: Challenges in Transitioning Software Project Management Methods and Strategies to Overcome Them

"In 2012, Dell software developers began transitioning from Waterfall to Agile (Scrum). O'Dwyer (Programme Manager) explained that a few years later, in summer 2016, the SPI Board decided that Dell software teams should adopt a newer set of Agile tools (collectively referred to as Flow), as it was gaining popularity in the global software community."

On the day Ferreira (Project Manager and Flow Coordinator) volunteered to add Flow coordination to his workload, O'Dwyer thanked him and said: **"It will take time to fully embed Flow changes into the team; change will be a challenge."** A passing colleague paused to comment: "Flow aims to produce financial savings. People may wonder: Will we work ourselves out of a job? There's a challenge for you!"

Another time, O'Dwyer mentioned that some Dell teams combined Scrum and Waterfall techniques, "which works really well for them." He added: "Perhaps some teams should combine Scrum and Flow." Ferreira replied: "That would be a good question to put to my NUI

Galway professors." Later that day, Ferreira spoke with Jimmy Delaney, a Senior Software Developer, to learn about his team's hybrid approach. Delaney explained,

> "We only blend Waterfall and Agile at the beginning of a project. Our experience is that successful projects use the Agile frequent delivery approach (two-week sprints), after first getting a robust preliminary definition of the expected final product or customer experience. I believe Flow will benefit the organization end-to-end. It will take time. How we nurture interest and get people really engaged – that's a challenge in itself."

Next, Ferreira spoke with Project Manager Pierre Dubois, who told him that Team 2 worked on backend order-processing systems, including an internal web-based sales tool. Dubois said: "People depend on scrum masters [team project managers] to show how to use Flow tools and explain basic principles, such as what work is in process or prioritized. Everyone needs training on Flow tools and on how to interpret Flow metrics."

Questions:

1. Why do you think O'Dwyer said, 'It will take time to fully embed Flow changes into the team; change will be a challenge'? What specific factors might make the transition to Flow difficult for Dell Software teams?
2. How can Dell teams effectively overcome the challenges of transitioning to Flow, considering both the technical and organizational aspects of the change?

Part 2: Scrum vs Flow

Scrum
Scrum consisted of four ceremonies:

1. Sprint Planning: Work was divided into two-week or four-week "sprints" (at Dell, two-week sprints were the norm).
2. Daily Scrum (or Standup): Each day, each team member reported to their scrum master and teammates about what they did the previous day and intended to do this day.
3. Sprint Review: At the end of each sprint, team members demonstrated new software code to stakeholders.
4. Sprint Retrospective: Also at the end of each sprint, the team gathered to reflect on how things went and what they would like to change.

A Sprint Product Backlog – a prioritized list of tasks for a team to work on next (e.g., features to support, bugs to fix) – was derived from a Roadmap and its requirements.

O'Dwyer focused on two agile software development performance indicators: user stories per project and user stories per two-week sprint (a.k.a. "velocity"). Regarding this latter metric, O'Dwyer told Ferreira that Dell software teams typically completed 12 to 15 user stories per sprint.

Flow
Flow, inspired by lean manufacturing (a waste-reduction/quality improvement technique), supported continuous software delivery. A promotion plan for early Flow books explained that developers could "achieve breakthrough quality, savings, speed, and business value by adhering to lean principles that ... already revolutionized manufacturing and R&D. Flow described how development tasks moved through a system. In a system with good Flow, work moved through steadily and predictably. In a system with bad Flow, work got stuck and started frequently. Thus, Flow supported software development by emphasizing continuous movement of value-added work, instead of discrete activities performed by separate teams or departments, through the entire development process. It differed from traditional software project management, in its focus on managing queues instead of project timelines and phases. Flow relied on two key quality metrics:

1. Cycle Time: How long a task was spent in specific individual or combined workflow states.
2. Lead Time: How long it took for a work item to move from initial request to delivery to a customer.

… O'Dwyer explained, he and other directors recognized that Flow was both agile and lean; similar cycle time and lead time metrics would help managers monitor software development productivity.

Further reading: Dennehy, D., & Conboy, K. (2018). Identifying Challenges and a Research Agenda for Flow in Software Project Management. Project Management Journal, 49(6), 103-118. https://doi.org/10.1177/8756972818800559

Question:

1. Why do you think the SPI Board mandated Dell teams to transition from Scrum to Flow? How does Flow address challenges or limitations in Scrum, and in what ways might it be more effective?

Appendix 1

Difference between Scrum and XP

source: seminar slide & https://www.geeksforgeeks.org/difference-between-scrum-and-xp/

Aspect	**Scrum**	**Extreme Programming (XP)**
Focus	Product backlog is broken into a Sprint Backlog and delivered as a product increment	User Stories are broken down and delivered as small, working features (story delivery)
Iteration Length	Sprints last 2–4 weeks	Iterations are shorter, lasting 1 week
Roles	Product Owner, Scrum Master, and Development Team	Customer, Developer, Tracker (metrics and progress), and Coach (XP expert)
Meetings	Includes Daily Scrums (~15 minutes), followed by Sprint Reviews and Retrospectives	Includes Weekly Cycle (progress review) and Quarterly Cycle (product release planning)
Flexibility in Timelines	Scrum models do not allow changes in their timeline or their guidelines.	Extreme Programming allows changes in their set timelines.
Emphasis	Scrum emphasizes self-organization.	Extreme Programming emphasizes strong engineering practices
Development Sequence	In the Scrum framework, the team determines the sequence in which the product will be developed.	In Extreme Programming, the team has to follow a strict priority order or pre-determined priority order.
Framework Adaptation	The Scrum framework is not fully described. If you want to adopt it then you need to fill the framework with your framework methods like XP, DSDM, or Kanban.	Extreme Programming(XP) can be directly applied to a team. Extreme Programming is also known for its **Ready-to-apply** features.
Software Engineering Practices	Scrum does not put emphasis on software engineering practices that developers should use.	Extreme Programming (XP) emphasizes programming techniques that developers should use to ensure a better outcome.

Engineering Methods	It requires developers to be conscious of adopting engineering methods to ensure better progress or quality.	It is very strict in adopting engineering methods such as pair programming, simple design, restructuring to ensure better progress or quality.
Feature Prioritization	In the preference of features, demand and priority do not have to be in line with one another.	In the preference of features, the demand corresponds to the priority.
Task Prioritization	In scrum, the scrum master asks the owner of the product to prioritize the tasks according to their requirements.	In XP, customer decides the job priorities being the owner of the product and then analyses the releases.
Flexibility in Task Prioritization	The tasks are prioritized by the owner of the product but with the flexibility that the priorities can be changed later on by the development team if required.	The tasks are prioritized by the customer and the task priorities cannot be changed by the development team.
Core Values	**Values-** · Openness · Focus · Commitment	**Values-** · Communication · Simplicity · Feedback
Customer Involvement	Customer involvement is less in the project.	Customer involvement is more in the project.