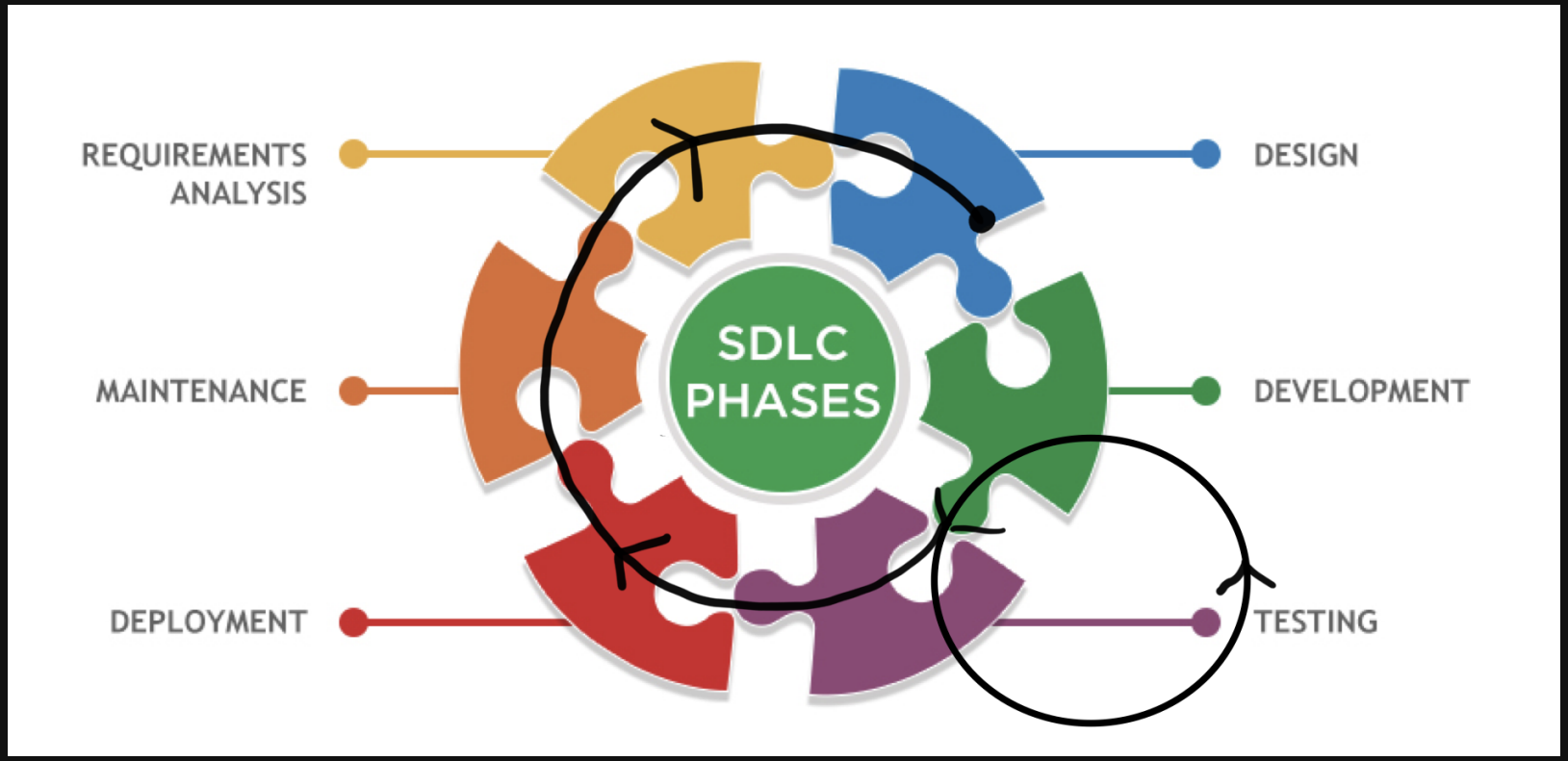
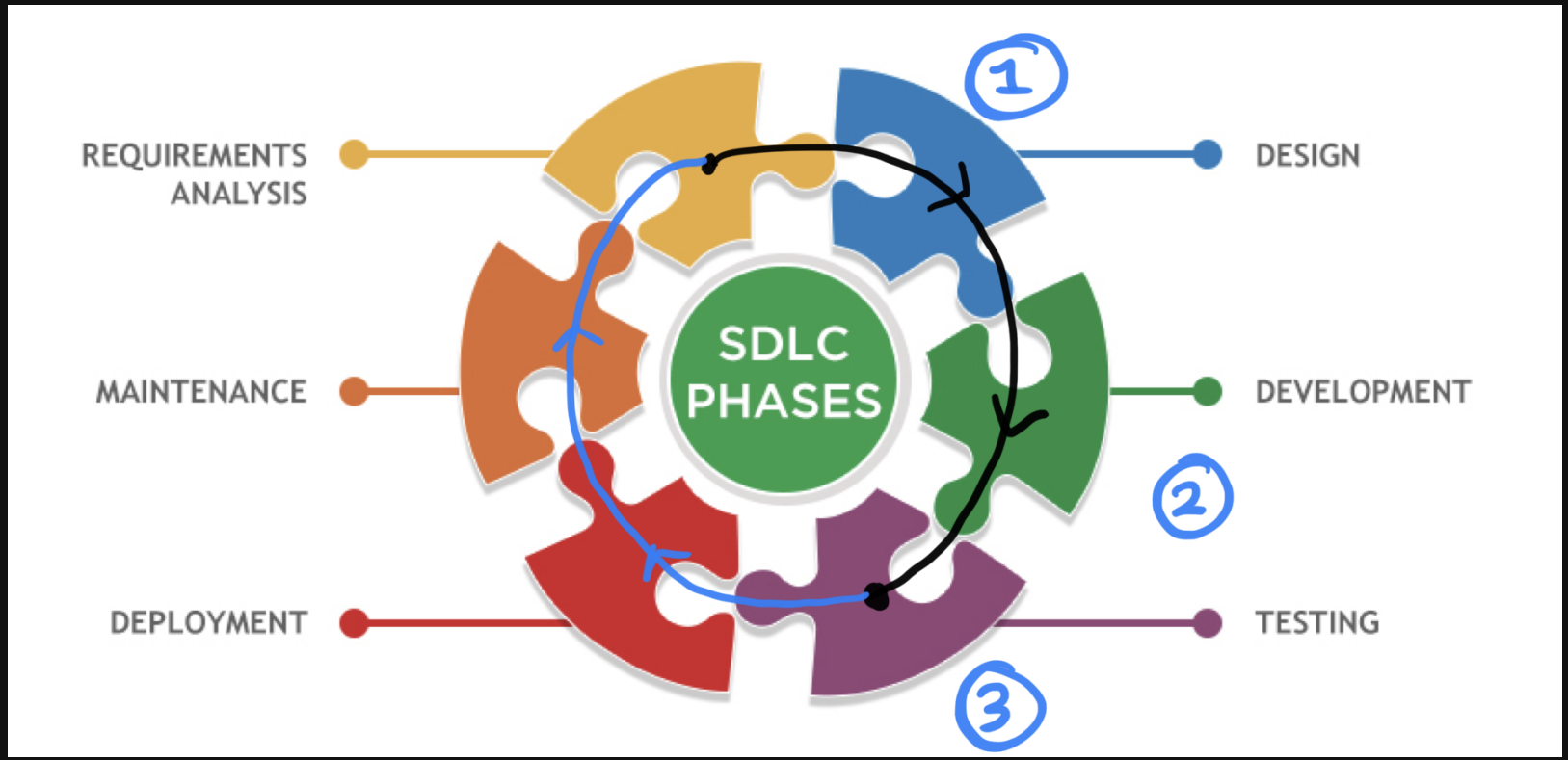
Agile Software Development & The Project

COMP2511



The Software Development Lifecycle


REQUIREMENTS ANALYSIS
DESIGN
MAINTENANCE
DEVELOPMENT
DEPLOYMENT
TESTING
SDLC PHASES
1
2
3

Requirements Analysis

User Stories & Epic Stories

- Role, Goal, Benefit
  - As a resident in a COVID hotspot, I want to be alerted of all new cases so that I can keep those around me safe
- **What's a good user story?**
  - Specific role
  - Singular goal
  - The benefit is actually beneficial to the user
- Epic Stories
  - Abstract user story that encompasses several stories
  - Describes user problems at a higher level

Acceptance Criteria

- Descriptive (Yes/No)
  - The user is given an alert whenever there is an update to restrictions
  - The user can see the number of active cases in the community
- Scenario-Based
  - Given, When, Then
  - **Given** that the user has scanned in to a venue, **when** there is a case identified at the venue, **then** the user is informed

Story Points and Priorities

- Story Points
  - Relative measure of **effort** needed to complete a user story
  - Includes time taken to research how to do it
  - Fibonacci Scale - 1, 3, 5, 8, 13, 21
- Priorities
  - High Priority - Minimum Viable Product (MVP)
  - Medium Priority
  - Low Priority

Example Gitlab Taskboard

https://gitlab.cse.unsw.edu.au/z5169779/taskboard-example-2511/-/boards

Agile Project Management

Timeline

- Using Story Points and Priorities, create a rough plan of how work will be completed
- Considerations
  - Sequencing of Tasks
  - Allocation of Tasks
  - Timespan of Tasks

	Monday	Tuesday	Wednesday	Thursday	Friday
Week 1	*Sprint planning meeting*	*Standup* Draft Interview Questions	*Standup* Review Interview Questions	*Standup* Conduct Interviews	*Standup* Collate Interview Results
Week 2	*Standup* Construct User Stories Draft UI Design	*Standup* Review UI Design Review User Stories	*Standup* Create Domain Model for Backend Decide on Software Architecture	Braedon away Validate User Stories	*End of sprint discussion*
Week 3	*Standup* *Sprint planning meeting*	*Standup* Finalise Technical Design	*Standup* Setup Database Create UI/UX design draft	*Standup* Create key entity stubs	*Standup* User Story 1: Alerts User Story 2: Check-ins

Braedon
Nick

Meeting Minutes

- Attendees
- (Optional) Agenda
- Discussion Points
- Action Items

Meeting 26/6

Attendees

- Nick
- Braedon
- Chloe
- Simon

Discussion Points

- LinkedServerCreator defect currently blocked due to concurrency problems in the main database
  - Locks to be placed by LinkedServerClass on main db while login is occurring
  - Need to check performance
- Audit Subscriber refactor
  - Usability review failed, running the subscribers accidentally deletes the subscribers which aren't in the same namespace
  - Simon's work to move the subscribers into their own namespaces needs to be merged into master before we can do anything else

Action Items

- Simon: Complete subscriber namespace movement and submit MR for review
- Chloe: Pull Simon's branch and continue refactor
- Nick: Lock main db when LinkedServer Class is logging in
- Braedon: Check performance of locking main db for login time (~5 seconds??)

Agile Practices

- Standups
- Asynchronous Standups
- Pair Programming

Test-Driven Development (TDD)

Why test first?

- Start with the end
- Tests-last simply justify the code written
- You want your tests to **fail** your code, not to **pass** your code

TDD with Java & Classes

1. Design your model (UML Diagram)
2. Write the class & method stubs
3. Write unit tests
4. Run the unit tests
5. Implement the functions
6. Pass the tests

Git Practices

- Tasks are assigned to team members;
- Tasks are updated across the kanban columns;
- The board shows the truth of your team's progress;
- The timeline you created in Milestone 1 is updated as needed in Milestones 2 and 3
- Assignment of tasks in the timeline corresponds to the board;
- Detail and specificity in commit messages;
- Avoiding committing large chunks of code;
- Merge requests are approved by another team member;
- One feature = one branch = one merge request into master.

Good Assumptions

- Clear up an ambiguity in the specification
- A good assumption articulates a **behaviour** rather than an **implementation**
- Assumptions are black-box