SCRUM

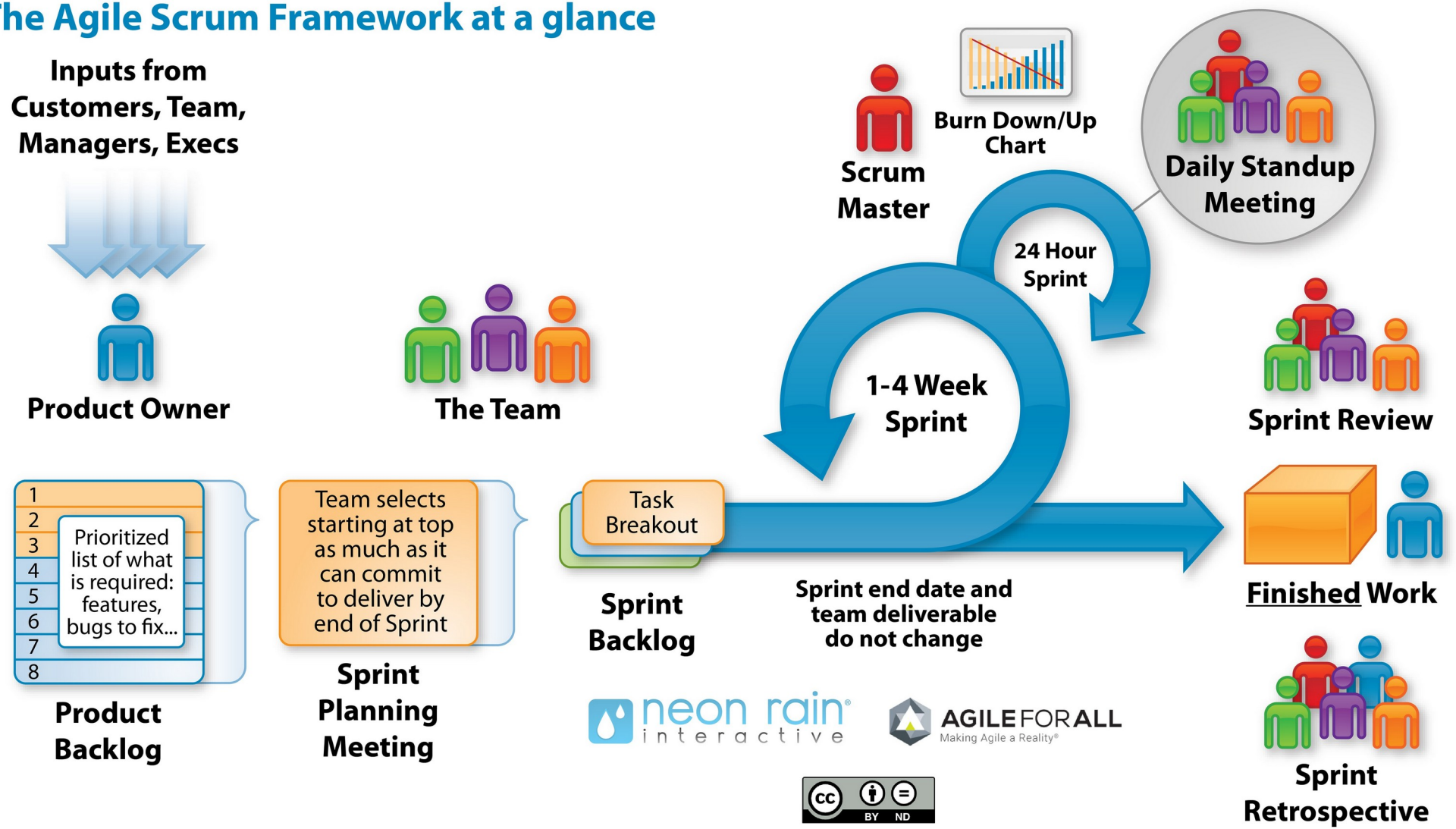
Module 4: Ceremonies I

Overview

- This is the first of several modules on working with ceremonies
  - We will start with Grooming and Planning ceremonies
  - How to define and achieve a clear sprint goal.
  - Setting the goals of each ceremony and defining expected outputs.
  - Structuring and managing ceremonies (time-boxing, facilitation techniques).
  - Specific tasks during the ceremony (e.g., how to split stories, update estimates).
  - Identifying recurring issues and measuring ceremony effectiveness.
  - Clarifying who should participate and in what role.
- The first part of today will be an overview how this is done
- The second part will be you practicing some of the ideas in groups

The Scrum Framework Reminder

The Agile Scrum Framework at a glance



The Increment

- Scrum uses very specific definitions for some of its terms
- What we are building in a Sprint is one of more increments
  - An Increment is a concrete step toward the Product Goal.
  - Increments are cumulative
    - *Each increment is added to all prior Increments*
    - *Each increments is verified, usually by passing a set of unit test.*
    - *The build that results from the increment is also integration tested ensuring that all Increments work together.*
  - Properties of an increment
    - *Usable: It can be put into use immediately, not necessarily in a production environment, but we can use the resulting product*
    - *Cumulative: Builds upon all prior increments, not just “this Sprint’s work.”*
    - *Transparent: Everyone can see the current state of the product.*
    - *Meets DoD: All completed items must align with the Definition of Done.*

The Definition of Done (DoD)

- Formal description of the state of the Increment
  - Happens when it meets the QA standards defined for that increment.
- DoD does not mean “finished coding” or completing a task
  - We define measurable success criteria for the test to pass.
  - These are generally referred to as acceptance tests
- There are generally two kinds of acceptance tests
  - Functional acceptance tests
    - *“When an valid login and Id is entered, the welcome page should be displayed”*
    - *Functional unit tests are run on the build that results from the increment*
    - *This generally uses some sort of test automaton tool.*
  - User acceptance tests
    - *Domain experts and stakeholders provide feedback. Usually qualitative.*
    - *Catches the details that the functional tests can’t, like the layout being difficult to understand*

The Definition of Done (DoD)

- The DoD of an increment will often include the following steps
  - Acceptance tests are added to test the functionality introduced by the increment
  - For the components to be build for the increment, unit tests are derived from the acceptance tests
    - *Eg. For the login feature mention in the previous slide, if we are building the component that validates the passworld*
    - *We take the different acceptance tests and determine what our component needs to do to make the acceptance tests pas*
    - *For example, we should reject a blank password or a password that is too showre*
  - Code is written, peer-reviewed, and unit tested successfully
  - Code builds without errors or warnings.
  - Integration (acceptance) tests are run, and no major defects remain.
  - Feature is documented (user-facing or technical).
  - Meets agreed acceptance criteria.
  - Approved by Product Owner (or meets business acceptance).
  - Deployed to staging/QA environment.

The Definition of Done (DoD)

- Scrum only says we need a measurable DoD
  - We define the measurable steps
  - For example, code review by code security specialists
- DoD requires three inputs
  - The Product Owner and Stakeholders
    - *They define what good enough is for the business to find the increment acceptable*
    - *They determine the pass/fail criteria from a business perspective*
  - The Developers
    - *They ensure that any item going into a Sprint backlog can be implemented as a feature*
  - The QA function
    - *Ensure the test case development, automation and execution are done according to QA bast practices*

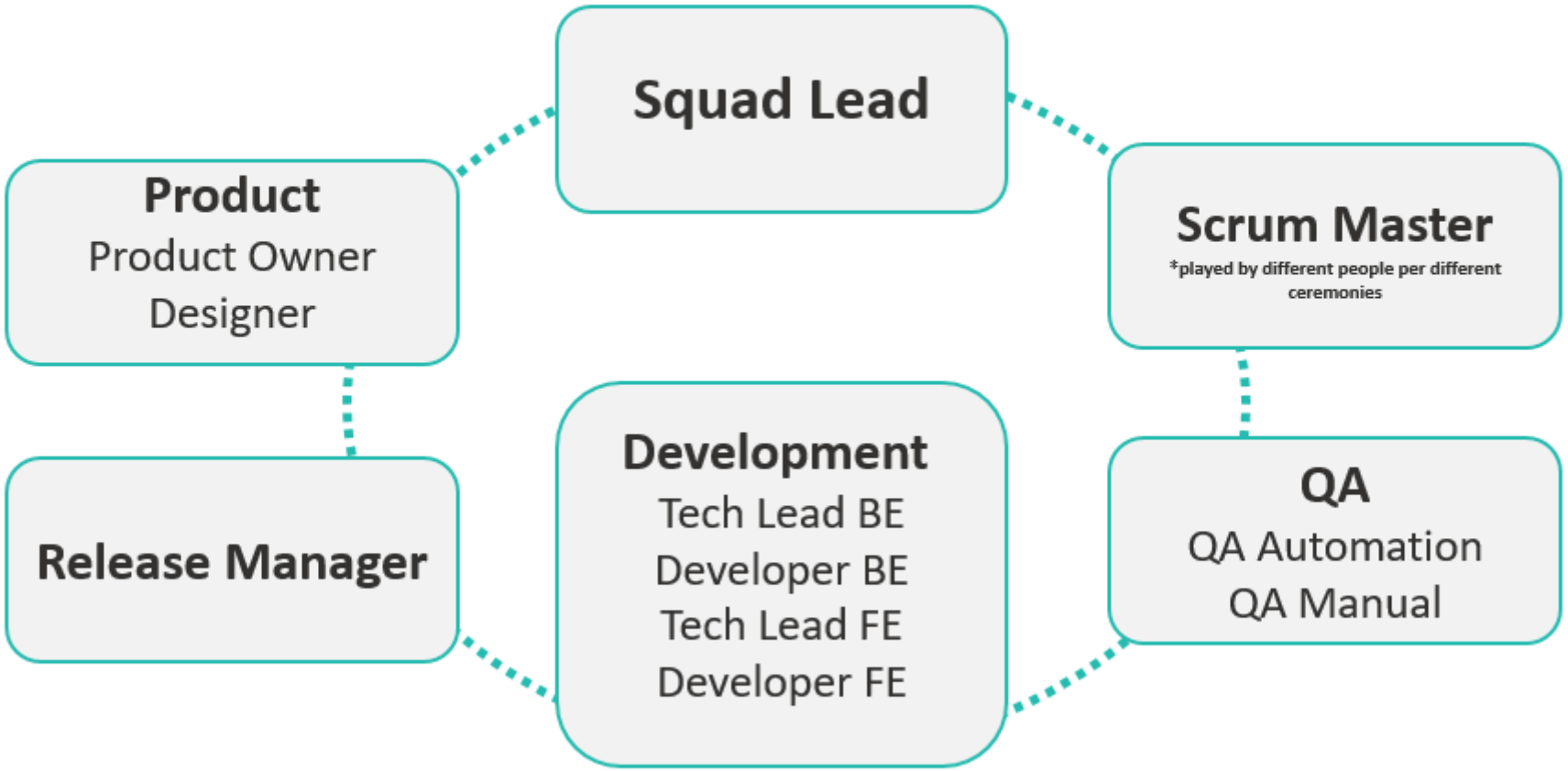
The KMS Example

- These are extracts from some of the KMS work
- The epic is “Add PostgreSQL support to KMS”
  - The documents is in the repository
  - One of the tasks you will do in your group is to establish an acceptance criteria so that we will know when PostgreSQL has been successfully added.
  - We will examine the documents at the start of the group work.

Team Structure

14

KMS Scrum Team Roles



KMS Lighthouse

The KMS Example

- These are extracts from some of the KMS work
- The epic is “Add PostgreSQL support to KMS”
  - The documents are in the repository
  - One of the tasks you will do in your group is to establish an acceptance criteria so that we will know when PostgreSQL has been successfully added.
  - We will examine the documents at the start of the group work.
  - Suggestion for DoD:
    - *“All of the queries that are currently executed in other database environments now execute correctly on PSQL”*
    - *Not the only one.*

Product Backlog Grooming

- Officially
  - Product Backlog refinement is the act of breaking down and further defining Product Backlog items into smaller, more precise items.
  - This is an ongoing activity in which the Product Owner and the Developers collaborate on the details of Product Backlog items.
- Examples of some specific tasks
  - Clarifying items
    - *Discuss items with the Product Owner and stakeholders.*
    - *Ensure developers understand the “why” and “what.”*
  - Breaking down items
    - *Large items (epics) split into smaller, testable stories.*
    - *Stories sized to fit comfortably in a Sprint.*
  - Prioritizing
    - *Product Owner orders items by business value, urgency, and dependencies.*
    - *Team provides input on technical risk and effort.*

Product Backlog Grooming

- Estimating
  - *Developers assign effort (story points, t-shirt sizes, etc.).*
  - *Estimates help the Product Owner make informed prioritization decisions.*
- Adding acceptance criteria
  - *Define clear conditions of satisfaction for each item.*
  - *Helps ensure transparency on “what Done means” for the story.*

- We will return to product backlog grooming in another session

Product Statement

- From the provided document
  - Multilanguage support was added only for MySQL databases in 7.6 MR3 and PostgreSQL support in KMS was suspended.
  - For VTB customer we need to fix PostgreSQL support.
  - Also, we plan to migrate all our environments from MySQL to PostgreSQL because of MySQL technical limitations.
  - The migration itself is not in story scope but all migration tools should be created and tested.
- Steps specified
  - Fix native KMS SQL queries to support PostgreSQL
  - Add PostgreSQL support to *https://kmsgit.lighthouse-cloud.com/kms/vanilla-db* repository
  - Make database switch easier using default KMS profiles
  - Make cloud scheduler setup easier using default KMS profiles
  - Fix KMS start from vanilla on PostgreSQL dump from *https://kmsgit.lighthouse-cloud.com/kms/vanilla-db* repository
  - Create migration tool from MySQL to PostgreSQL and test it with big existing production dumps (see related task: *https://kms-lighthouse.atlassian.net/browse/LAS-7104)*

Product Statement Acceptance Criteria

- Acceptance Criteria
  - A new KMS PostgreSQL dump is created in https://kmsgit.lighthouse-cloud.com/kms/vanilla-db repository on each push to dev branch in https://kmsgit.lighthouse-cloud.com/kms/kms-trunk or https://kmsgit.lighthouse-cloud.com/kms/item-uploader repository
  - KMS database vanilla instance could be easily created with docker images. The server image could start PostgreSQL with KMS dump inside. The client image could push KMS dump into external PostgreSQL instance.
  - QA automation tests passed with PostgreSQL configuration
  - Regression tests passed successfully on QA environment
  - Regressions tests passed after MySQL->PostgreSQL migration for a big customer
  - When story is done we need to create additional story to remove MySQL, Oracle, and MSSQL support from KMS

Product Statement Acceptance Criteria

- Operational definitions
  - An operational definition is a measurable definition used to quantify something we need to measure
  - For example: *The interface is easier to use*
    - What does easy mean? How would we know we made the interface easier to use
    - Otherwise we don't have a specific goal and wind up with scope creep
  - We create an operational definition of what "easy to use" means
    - Identify specific issues that users find difficult to use
    - For example, location of items is difficult to use in a workflow
    - Or, the labeling of items is misleading or inconsistent with the terminology users normally use
    - Or, the workflow for a task is confusing or overly complex
  - This is where we have consultation with the PO and stakeholders
    - Identify the problematic issues
    - Define the acceptance criteria that would verify that we made that we resolved that issue

Sprint Planning

- Input: Product Backlog
  - Product Owner presents the highest-priority items (those that are refined, clear, and meet Definition of Ready).
  - Items are aligned with the Product Goal and business priorities.
- Sprint Planning Meeting
  - Time-boxed, usually up to 8 hours for a one-month Sprint
  - Has three key parts:
  - Why is this Sprint valuable? (Sprint Goal)
    - *Product Owner and Developers collaborate to set a Sprint Goal that gives the Sprint purpose.*
  - What can be done in this Sprint? (Select items)
    - *Developers assess the Product Backlog items and select those they believe can be completed within the Sprint.*
  - How will the chosen work get done? (Plan work)
    - *Developers break selected PBIs into smaller tasks (not mandatory, but common practice).*

Sprint Planning

- Additionally
  - We look for items that other items will depend on

Acceptance Criteria

- When developing acceptance criteria, the IEEE best practices are a useful guide

Complete	*The acceptance tests cover all possible inputs and conditions, both valid and expected as well as invalid and unexpected.*
Consistent	*No two acceptance tests require to system to behave differently when the state and input of two tests are the same.*
Correct	*The expected result of each test meets the acceptance criteria.*
Testable	*All test inputs, states and outputs are quantified and measurable.*
Verifiable	*There is a finite cost effective process for executing each test.*
Unambiguous	*There is only possible way to interpret or understand each test and test result.*
Valid	*Everyone can read, understand, and analyze the tests well enough to formally approve the tests.*
Modifiable	*The test cases are organized in a way so that they are easy to use, modify and update.*
Ranked	*The test cases are in a priority order that everyone on both the business and technical sides agree on.*
Traceable	*Every test case can be traced back to the example and acceptance criteria that motivated it and then back to the original requirement.*

Sprint Planning Exercise 1

- First Step
  - Given the following steps, decide which should be addressed first
    - *Fix native KMS SQL queries to support PostgreSQL*
    - *Add PostgreSQL support to https://kmsgit.lighthouse-cloud.com/kms/vanilla-db repository*
    - *Make database switch easier using default KMS profiles*
    - *Make cloud scheduler setup easier using default KMS profiles*
    - *Fix KMS start from vanilla on PostgreSQL dump from https://kmsgit.lighthouse-cloud.com/kms/vanilla-db repository*
    - *Create migration tool from MySQL to PostgreSQL and test it with big existing production dumps*
- For the choice
  - Identify why it should be done first?
  - What sort of refinement is necessary for us to define a Sprint Goal?
    - Clarification of terms – quantify terms that are vague or open for interpretation
    - Quantify the specific items to be addressed
    - What is the scope of the specific sprint?
    - What sort of acceptance criteria would we use for our DoD?
    - What additional information do you need to help make the choice?

Questions