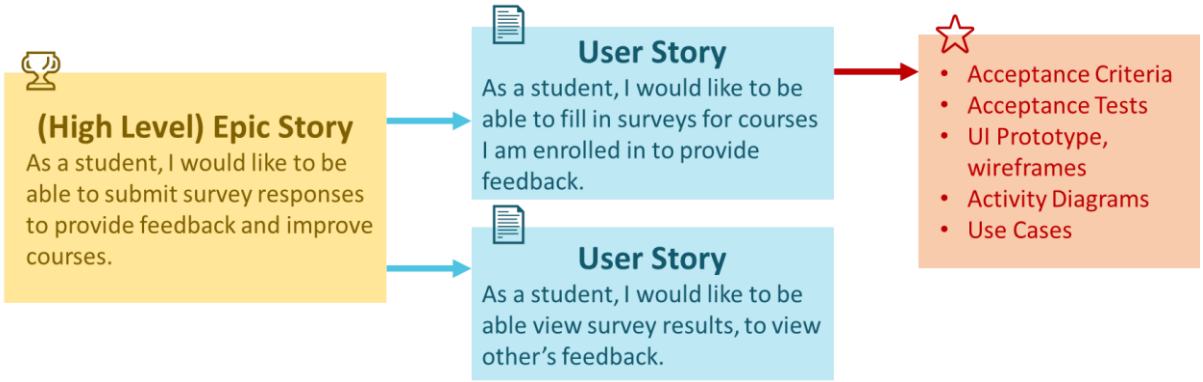
3. Requirements Engineering

COMP1531 • KC Notes

3.1 Requirements Engineering

- Requirements engineering: **formulating a well-defined problem to solve**
  - Has a **set of criteria**, where problems solve or fail to solve the criteria
  - Involves different stakeholders – end users, business owner, architects and developers
- **Gathering**: understand business context, gather what is required, to be accomplished
- **Analysis**: refining and identifying dependencies, conflicts, risk, ensures developer understanding matches customer expectation
- **Specification**: Documents the function, quality and constraints of the software
  - UML use-cases (scenarios describing how end user interacts)
  - System Requirements Specification (SRS)

3.2 Agile Requirements Engineering



- Start with **visioning**: identify **epic stories**, key features and target users
- Brainstorm and breakdown features into **user stories**
- Detail user stories to yield **iteration deliverables**
- **User stories**: short simple descriptions of a feature narrated from the perspective of the person who desires that capability – Role, Goal, Benefit
  - Assign **unique identifier**
  - Estimate **size (time length, measured in story points) and priority**
  - Remember **non-functional requirements** (e.g. online via a browser)

As a <type of user>, I want <some goal> so that <some reason>.

- **Three C's Model**: card (physical token), conversation (conversation with different stakeholders), confirmation (formal confirmation of acceptance criteria)
- **INVEST**: evaluates the quality of a user story
    - **Independent**: can be developed independently and **delivered separately**
    - **Negotiable**: discussable further
    - **Valuable**: reason is clear
    - **Estimable**: understandable and can be estimated
    - **Small**: deliverable within an iteration
    - **Testable**: has clear acceptance criteria with **error conditions**

3.3 Acceptance Criteria

- **Acceptance Criteria**: Statements of requirements **described from the perspective of the customer** – **what is required** for the business owner to **accept the user story** as "done"
- **Types of Requirements**:
    - **Functional requirements** (inputs and outputs)
    - **Non-functional requirements** (security, usability, reliability, performance, supportability)
    - **On screen appearance requirements** – should be simple and hand-drawn

3.4 Use Case Modelling

- **Use case**: step by step description of **how a user will use the system-to-be** to accomplish business goals
    - An envisioned sequence of actions and interactions between actors and the system

Actor	Actor's Goal	Use Case Name
Student	To choose and type responses in a form	AnswerSurvey (UC-1)
SurveySystem	To save a student's responses	SaveAnswers (UC-2)
SurveySystem	To light up questions where the student has incomplete answers in the survey	IncompleteAnswers (UC-3)

- Types of actors:
    - **Initiating actor**: user who initiates the use case to achieve a goal
    - **Participating actor**: user who participates but does not initiate
        - Supporting actor: helps complete the use case
        - Off-stage actor: passively participates the use case, e.g. Student in UC-3
- Use cases can be generalised, e.g. 'UC-4: Complete survey'
- Use cases can be **extended** – can be optional or conditional
- **Traceability Matrix**: mapping system requirements to use cases

Req't	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8
REQ1	5	X	X						
REQ2	2		X						
REQ3	5	X						X	
REQ4	4	X						X	

3.5 System Sequence Diagram

- Model a system workflow
    - Arrows should be able to be “followed”
    - Dashed arrows for returning a result (e.g. from a database)