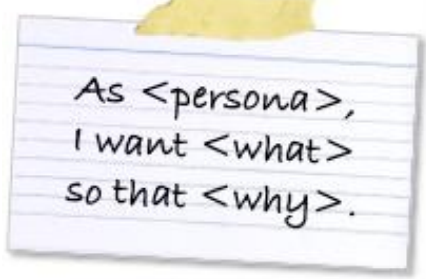
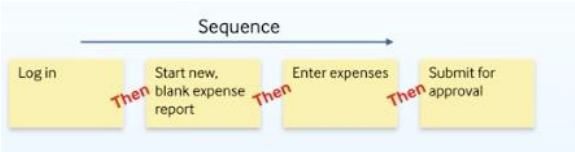
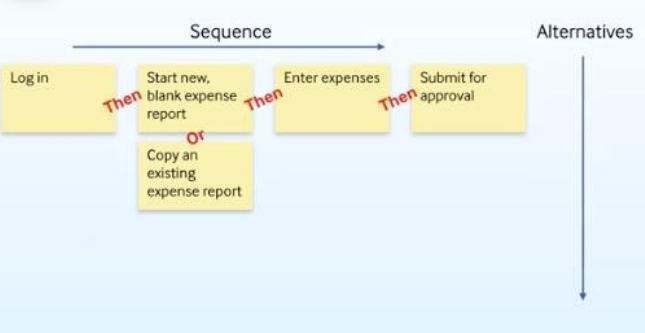
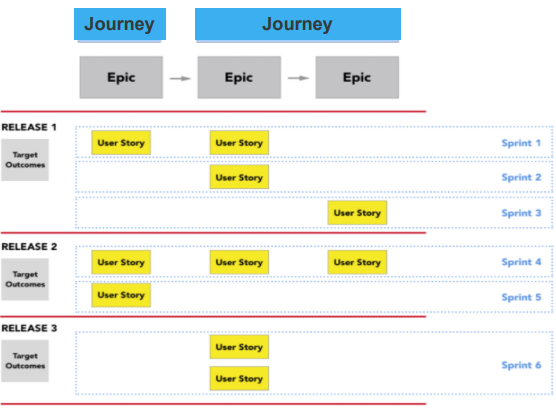
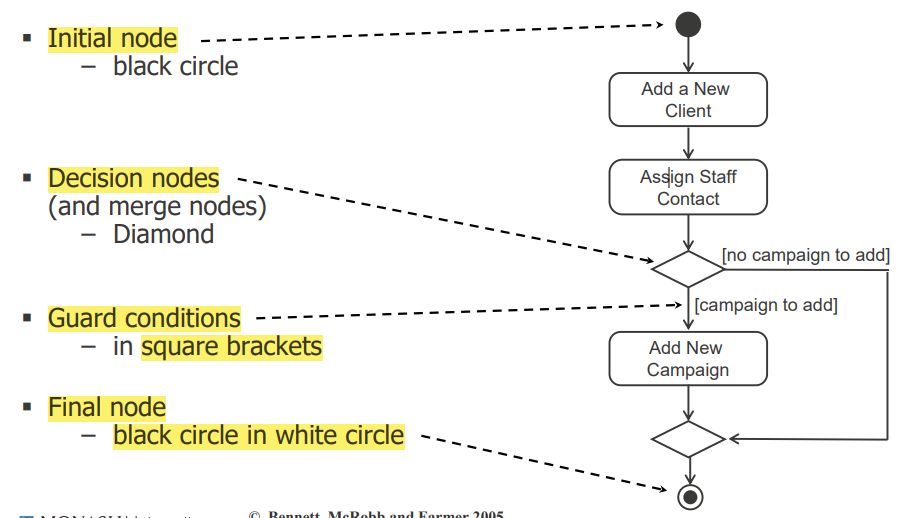
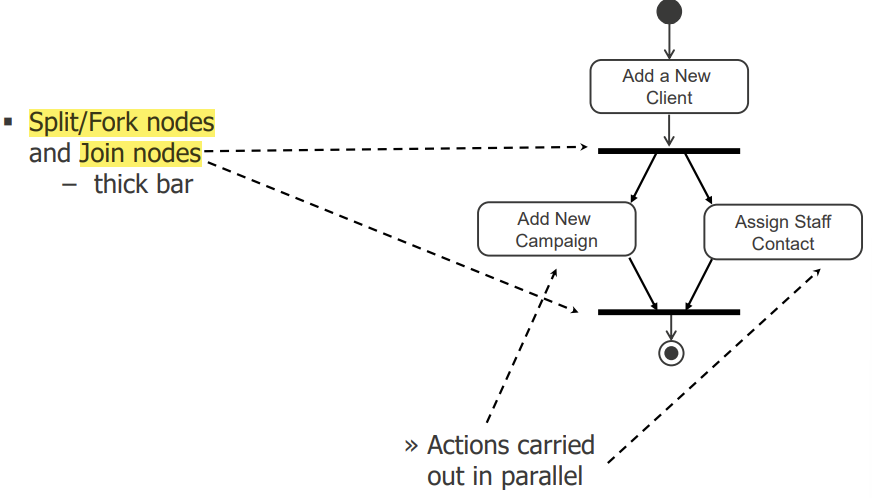
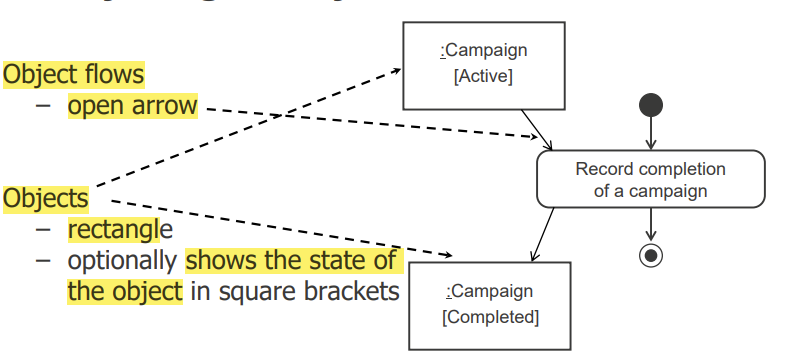
Reasons for modelling

* Reducing complexity’
* Communication with other development team
* Communication with stakeholders
* Communication with stakeholders
* Documenting all the details of requirements

Gathering requirements in Agile User Stories



* Short, simple description of a product feature
* go into the Product Backlog
* Why User stories
  + Encourages user communication
  + Focus on end user value
  + Planning is simplified – if it’s too big and you can’t estimate
  + Avoids locking in design detail too early
  + Users do not need to be trained to understand User stories
  + Eliminates weighty documentation
* Story Writing Workshops-continue with s3
  + Encourage the entire team to participate in the process
  + Running Workshops
    - Focus on a single objective
    - Map the big picture – brainstorm, list of things users do, cluster, remove duplicates
    - Add user stories
    - Visualise the relationship between stories - Story mapping
    - Map out releases
* How do you write
  + Card- written on index card, Annotated with note model and prototype
  + Conversation- detail come out with product owner
  + Confirmation-what is going to make user accept the story
  + Use a template
    - Define end user
    - Specify what they want
    - Describe benefit
    - Acceptance criteria
  + Identify the user type
    - Focus on the personas who interact with the system
    - Avoid the generic role "User”
  + Ask questions
  + get the size right
    - small enough for the development team
    - big enough to represent business value
    - able to be delivered in its own right
    - User stories that are too big, are harder to understand, estimate, and implement successfully
  + Story too large?
    - EPIC - a LARGE story that is split into smaller user stories
    - Product Backlog items tend to start as epics broken into smaller user stories during Sprint planning
  + adding the Acceptance criteria
    - a set of predefined requirements must be met to mark a user story complete
    - Reduces ambiguity
    - Prevents miscommunication
    - defines what you need to test to meet requirements
* How good is your User Story?
  + not dependent on other stories developed separately
  + leave room for negotiation
  + gives value to the customer
  + enough information to be estimated
  + small enough to fit within a sprint
  + includes acceptance criteria to test that customer needs met
* Common mistakes
  + Too formal / too much detail
  + Technical tasks impersonating user stories
  + Skipping the conversation
* Story Mapping
  + Helps arrange user stories into a useful model
  + Helps identifying holes and omissions in your backlog
  + ask questions
    - What will the user most likely want to do next?
    - What mistakes could a user make here?
    - What could confuse a user at this point?
    - What additional information could a user need?
  + Sequence 
  + Alternatives 
  + higher the value the higher up the story is placed 
* **Release planning**
  + a set of functionalities that makes sense to deliver to users at the same time
  + Create horizontal swim lanes for each release – move stories up or down
  + A Release can include multiple sprints
  + Release 1 – 4 user stories make sense to be given to users 
* **Activity diagrams-** describe procedural logic, business processes, and work flows
  + meaning/detail to our user stories
* symbols

**** **** 

* Drawing
  + Identify actions
  + sequential order
  + alternative flows and the conditions
  + Add decision and merge nodes, flows and guard conditions
  + Identify any actions that are carried out in parallel
  + Add fork and join nodes and flows
  + Identify any processes that are repeated
  + add swim lanes to show the responsibilities
  + Add the object flows and objects