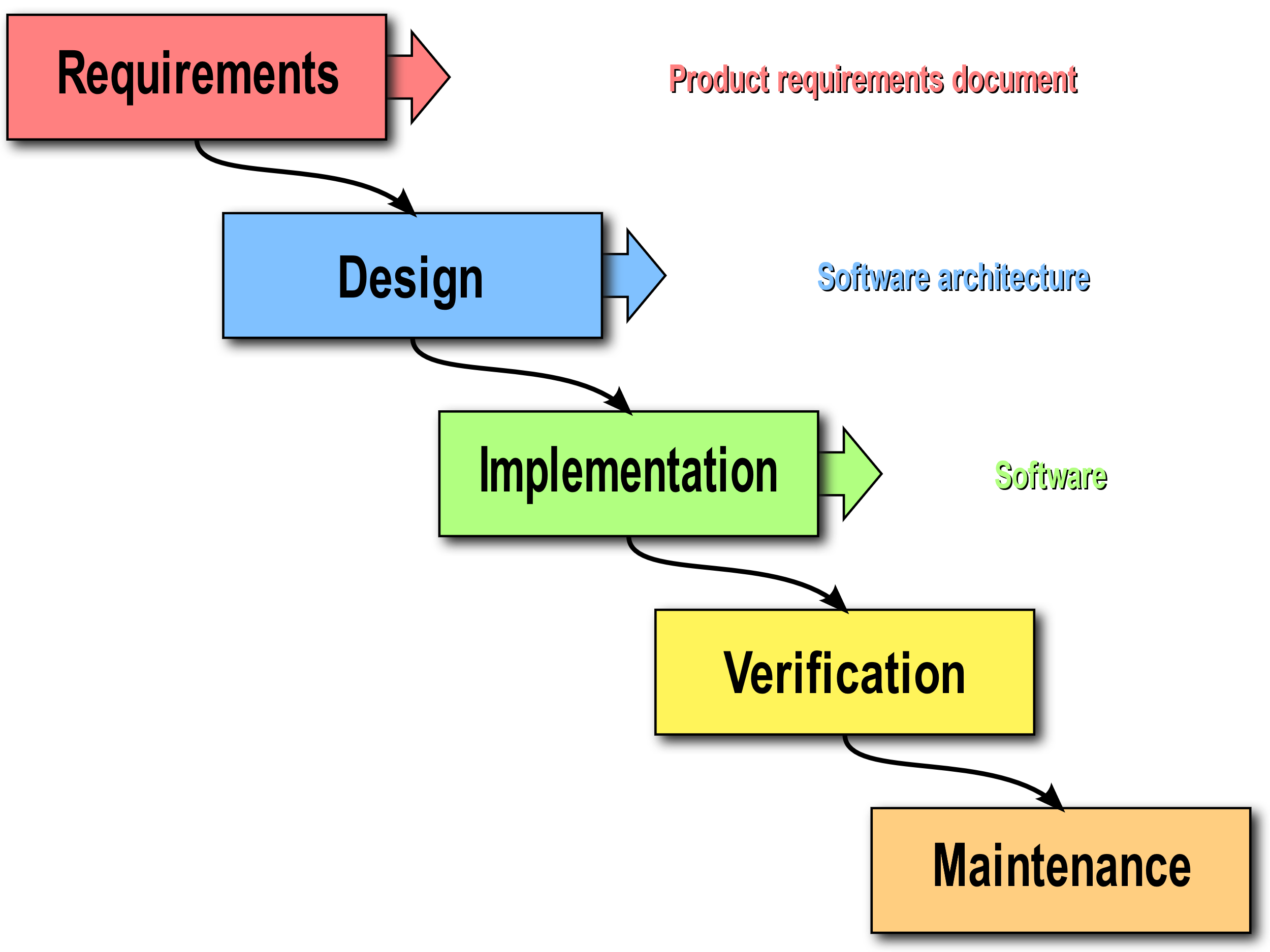
SDLC (Software Development Lifecycle)

Waterfall (Old School)



* Old(er) approach to software development
* You only go down
* There is a rigid strict adherence to the steps
* Cons to the waterfall methodology
  + Does not account for the lessons learned while developing
    - Halfway through the project you realize that the interface is poorly designed/named for what it is need for
  + Rigid adherence to the steps can slow progress
    - Your part of designing is done but you have to wait on someone else
  + Leads to hierarchical team structures where
    - It is not easy to reallocate people to squads that need help
  + Leads to fulfilling the contract over what the owner really wants
  + No working application until the very end of development

Agile Methodology of the (SDLC)

* Agile is a mindset. Not a particular set of steps to follow
* It gets the term Agile from the belief that all members of the development should readily adapt to the situation.
  + Implementations
    - Scrum
    - Kanban
* 4 main tenets of Agile
  + Individuals and interaction over processes and tools
    - Developers should be able to easily reach out and communicate with each other
      * Anti-example. A developer for the database cannot talk to the front-end developers directly. He has to send a formal email to the manager to request a meeting
  + Working Software and demos over documentation
    - Having a working product is far superior than the best notes ever written
    - The demo does not have to have full functionality
    - It is better to have an app where login is working but certain features when login are not as opposed to nothing working but it’s well documented
    - Do not take this as an excuse to not have any documentation
  + Customer collaboration over contract negotiation (requirements)
    - Always talk to the client/customer to get clarification on what they want
  + Responding to change over following a plan
    - Plans are worthless but planning is everything
    - As you code you will learn lesson about how best to design the application

Scrum (Implementation of Agile)’

From a rugby scrum

* Scrum Roles
  + Product Owner
    - Person who is ultimately responsible for the presenting the end client a working product
    - Lot of face to face time with the stakeholders
    - Highest person on a Scrum team
    - May or may not necessarily be a programmer
  + Stakeholders (Not really part of the team)
    - People who are contracting you to make the product
  + Scrum Master (technical leader)
    - Head of the Scrum Development team’
    - A very senior developer with significant experience
    - Their main role is the make the Scrum team work as efficiently as possible
      * Allocating tasks to the people most qualified to do them
      * Remove any blockers
        + Buy a software license if a developer needs it
      * Easily Approachable
      * Cheerleader
        + Keep people motivated and engaged in the project
      * Mentor
  + Team Members (Scrumling)
    - Developers
    - Business Analysts
      * People who guide development usually through creating user stories and communicating with the client

Sprint

* Incremental unit of work
* Typically its about 2 weeks long
* At the end of each sprint you should have something to demo

Sprint Phases

* Start (day 1)
  + Initial planning
    - Create user stories
    - Assign points to the user stories
      * Points are a way to assign difficulty
  + Create a spring backlog
    - Tasks to be completed during a sprint
  + Assign members to those tasks
  + Set a reasonable goal for what to accomplish by the end of the sprint
  + Usually done by the Scrum Master and senior developers
* Throughout (days 2-13)
  + **Daily Standup**
    - Everyone will communicate to the entire team how their tasks are progressing
      * Ask for additional help
      * Talk about any issues you are having
      * Mention if you have extra capacity
      * Task be reassigned because you cannot do it
  + Development/testing
    - Test as you develop
    - TDD!!!
* End (day 14)
  + Sprint retrospective
    - What was accomplished and what was not
    - Sprint metrics
      * How many user stories were completed
      * How many points were completed
    - Lessons learned
      * Propose design changes
      * Propose a new technology
    - Prepare a demo
  + Use you what you learned to plan the next sprint