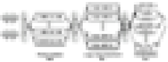
"Plan and Document": Before coding project manager makes plan. Write detailed documentation of all phases of plan. Progress measured against the plan. Changes to project must be reflected in documentation and possibly to plan.

	Waterfall	Spiral	Agile
Description	1. Requirements analysis & specification 2. Architectural design 3. Implementation & Integration 4. Verification 5. Operation & Maintenance Depends on project manager	Combine Plan and Document with prototypes. Rather than plan & document all requirements 1st, develop plan & requirement documents across each iteration of prototype as needed and evolve with the project	Constant iteration, Agile emphasizes: Test-Driven Development (TDD) to reduce mistakes, written down User Stories to validate customer requirements, Velocity (avg # of points/week) to measure progress
Benefits	Good for bigger teams	Involve prototypes & customers	Better iterations

HTTP:

- HTTP request includes request method (GET, POST, etc.), Uniform Resource Identifier (URI), HTTP protocol version understood by the client, headers—extra info regarding transfer request
- Since HTTP is stateless, you use cookies in order to store data to help guide them through the pages of the app (e.g. authentication, (fewer) tracking, customization, etc.)

3-Tier Shared Architecture



Shared Nothing Architecture

MVC:

- Controller mediates between model and view
- Any instance variable declared in the controller is made available to the view
- Subclassing from ActiveRecord::Base gives each model the ability to perform CRUD
- Any instance variable declared in the model that is not part of the database will be lost/unavailable in between every request made to the app
- In MVC, each interaction the user can do is handled by a controller action
- Routes: routes map <HTTP verb, URI>→controller action

Ruby:

- Reflection lets us ask an object questions about itself and have it modify itself
- Metaprogramming lets us define new code at runtime
- Modules = Collection of methods that aren't a class
- A closure is the set of all variable bindings you can "see" at a given point in time & "closes over" all the variables you can see, the binding inside of closure
- Ruby blocks are closures: they carry their environment around with them.
  - Result: blocks can help reuse by separating what to do from where & when to do it
- Duck typing encourages behavior reuse: "Quacks like" something and ask: "everything is a method call—do you respond to this method?"