“Plan-and-Document”

* Before coding, project manager makes plan
* Write detailed documentation all phases of plan
* Progress measured against the plan
* Changes to project must be reflected in
* documentation and possibly to plan

5 phases of Waterfall “lifecycle”

* Requirements analysis & specification
* Architectural design
* Implementation & Integration
* Verification
* Operation & Maintenance
* Complete one phase before start next one
* Why? Earlier catch bug, cheaper it is
* Extensive documentation/phase for new people’

Waterfall flaw: Customers see the product, but want changes.

Often after build first one,

developers learn right way

they should have built it

Spiral Lifecycle (1986)

* 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 schedule targets
* Iterations involve the
* customer before the
* product is completed
* Reduces chances of
* misunderstanding
* Risk management part
* of lifecycle
* Project monitoring easy
* Schedule & cost more
* realistic over time
* Iterations 6 to 24 months
* long
* time for customers to
* change their minds!
* Lots of documentation
* per iteration
* Lots of rules to follow,
* hard for whole project
* Cost of process is high
* Hard to meet budget and
* schedule targets

Project Managers: Write contract to win project, recruit dev team, evaluate performace/salary

Estimate costs, evaluate risks, document project management plan.

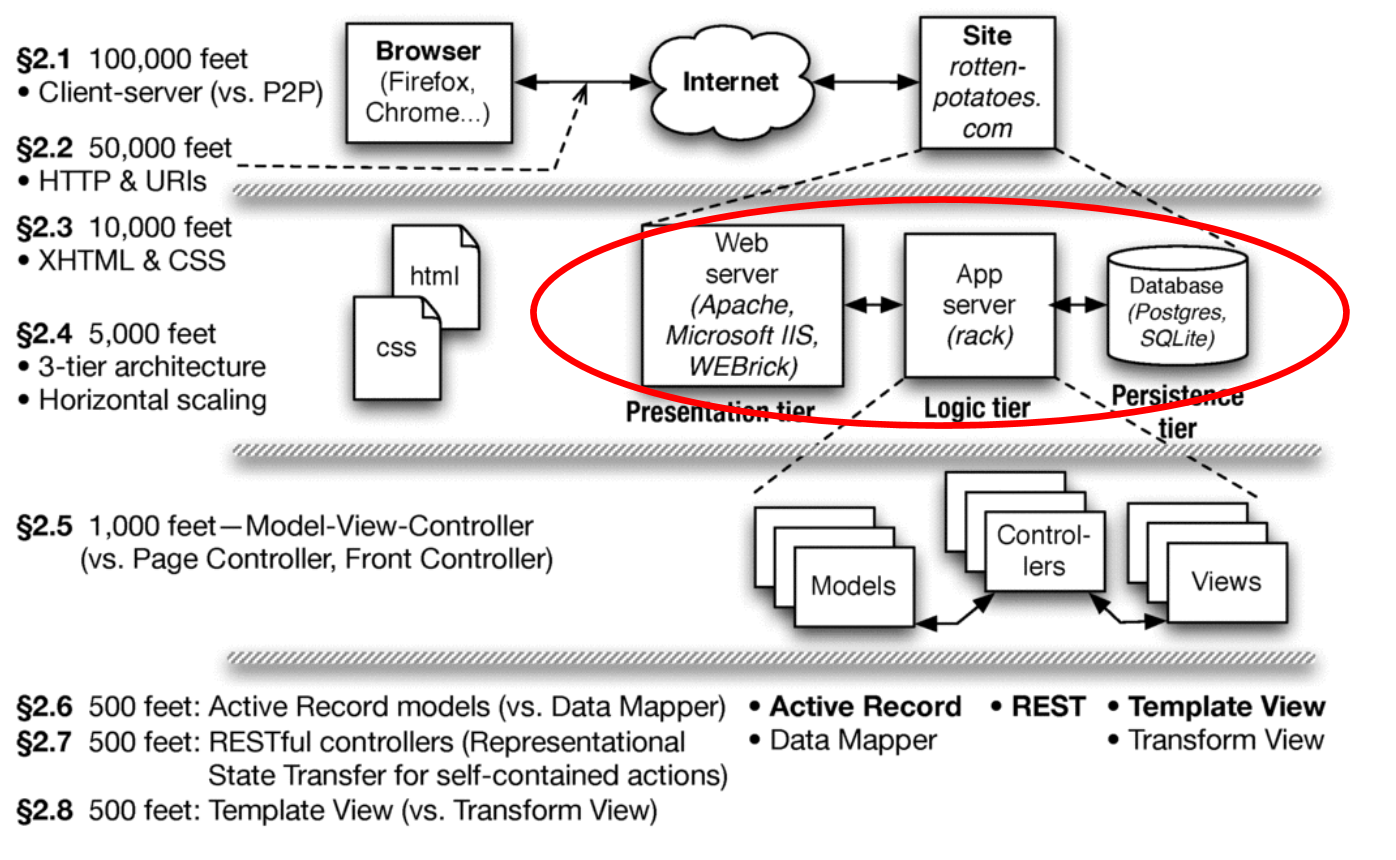
Agile Manifesto:: “We are uncovering better ways of developing SW by

doing it and helping others do it. Through this work we

have come to value

* Individuals and interactions over processes & tools
* Working software over comprehensive documentation
* Customer collaboration over contract negotiation
* Responding to change over following a plan
* That is, while there is value in the items on the right, we
* value the items on the left more.”
* If short iterations are good, make them as short as
* possible (weeks vs. years.
* If simplicity is good, always do the simplest thing
* that could possibly work
* If testing is good, test all the time. Write the test
* code before you write the code to test.
* If code reviews are good, review code
* continuously, by programming in pairs, taking
* turns looking over each other’s shoulders.

SaaS: Communication, Scalability, Dependability, Servers on clusters

To add a new action to a Rails app

* Create route in config/routes.rb if needed
* Add the action (method) in the appropriate
* app/controllers/\*\_controller.rb
* Ensure there is something for the action to
* render in app/views/model/action.html.haml
* We’ll do Show action & view (book walks
* through Index action & view)

Model: methods to get/manipulate data

Controller: make data from Model available to View

View: display data, allow user interaction

* Show details of a movie (description, rating)

GET - Requests data from a specified resource. POST - Submits data to be processed to a specified resource

Answers 3 questions at “daily scrums”:

What have you done since yesterday?

What are you planning to do today?

Are there any impediments or stumbling blocks?

Help individuals by identify what they need

ScrumMaster: team member who

Acts as buffer between the Team

and external distractions

Keeps team focused on task at hand

Enforces team rules

Answers 3 questions at “daily scrums”:

What have you done since yesterday?

What are you planning to do today?

Are there any impediments or stumbling blocks?

Help individuals by identify what they need

ScrumMaster: team member who

Acts as buffer between the Team

and external distractions

Keeps team focused on task at hand

Enforces team rules

Removes impediments that prevent team

from making progress

Product Owner: A team member (not the

ScrumMaster) who represents the voice of the

customer and prioritizes user stories

Start and stop meeting promptly

Agenda created in advance; no agenda, no meeting

Minutes recorded so everyone can recall results

One speaker at a time; no interrupting talker

Send material in advance, since reading is faster

Action items at end of meeting, so know what each

should do as a result of the meeting

Set the date and time of the next meeting

|  |  |  |
| --- | --- | --- |
|  | **[abc]** | A single character of: a, b, or c |
|  | **[^abc]** | Any single character except: a, b, or c |
|  | **[a-z]** | Any single character in the range a-z |
|  | **[a-zA-Z]** | Any single character in the range a-z or A-Z |
|  | **^** | Start of line |
|  | **$** | End of line |
|  | **\A** | Start of string |
|  | **\z** | End of string |

|  |  |
| --- | --- |
| **.** | Any single character |
| **\s** | Any whitespace character |
| **\S** | Any non-whitespace character |
| **\d** | Any digit |
| **\D** | Any non-digit |
| **\w** | Any word character (letter, number, underscore) |
| **\W** | Any non-word character |
| **\b** | Any word boundary |

|  |  |  |  |
| --- | --- | --- | --- |
| **(...)** | | Capture everything enclosed | |
| **(a|b)** | | a or b | |
| **a?** | | Zero or one of a | |
| **a\*** | | Zero or more of a | |
| **a+** | | One or more of a | |
| **a{3}** | | Exactly 3 of a | |
| **a{3,}** | | 3 or more of a | |
| **a{3,6}** | Between 3 and 6 of a | |

Creating a resource usually

takes 2 interactions

new: Retrieve blank form

create: Submit filled form

How to generate/display?

How to get values filled in by

user?

What to “return” (render)?

To create a new submittable form:

Identify the action that serves the form itself

Identify the action that receives submission

Create routes, actions, views for each

(may already exist if using “resource” style routes)

Form elements’ name attributes will appear as

keys in params[]

Helpers provided for many common elements

Creating the Form

Anatomy of a form in HTML

the action and method attributes (i.e., the route)

only named form inputs will be submitted

Generating the form in Rails

often can use URI helper for action, since it’s just the

URI part of a route (still need method)

form field helpers (see api.rubyonrails.org) generate

conveniently-named form inputs

Idiom: redirect user to a more useful page.

e.g., list of movies, if create successful

e.g., New Movie form, if unsuccessful

Redirect triggers a whole new HTTP request

How to inform user why they were redirected?

Solution: flash[]—quacks like a hash that persists

until end of next request

flash[:notice] conventionally for information

flash[:warning] conventionally for “errors”

Flash & Session session[]: like a hash that persists "forever"

reset\_session nukes the whole thing

session.delete(:some\_key), like a hash

By default, cookies store entire contents of session

& flash

Alternative: store Rails sessions in DB table

(Google “rails session use database table”)

Another alternative: store sessions in a “NoSQL”

storage system, like memcached

Don't do what customers want

Or projects are late

Or over budget

Or hard to maintain and evolve

Or all of the above

≥5 => divide user story into simpler

stories, group into epics

backlog not too demanding

Doesn’t matter if velocity is

5 or 10 points per iteration

As long as team consistent

Idea is to improve self-evaluation

and suggest number of iterations for

feature set

Specific and Measurable

Each scenario testable

Implies known good input and expected results exist

Anti-example:

“UI should be user-friendly”

Example: Given/When/Then.

Given some specific starting condition(s),

When I take specific action X,

Then one or more specific thing(s) should happen

Achievable

Complete in 1 iteration

If can’t deliver feature in

1 iteration, deliver subset of stories

Always aim for working code @ end of iteration

If <1 story per iteration ,need to improve point estimation per story

Relevant

Discover business value, or kill the story:

Protect revenue

Increase revenue

Manage cost

Increase brand value

Making the product remarkable

Can you include stories that don’t have obvious business value?

Timeboxed

Stop story when exceed time budget

Give up or divide into smaller stories or reschedule what is left undone

To avoid underestimatinglength of project

Pivotal Tracker tracks velocity, which helps avoid underestimate

