COMP30830
Software Engineering (Conversion)

Practicum
05/06/2019

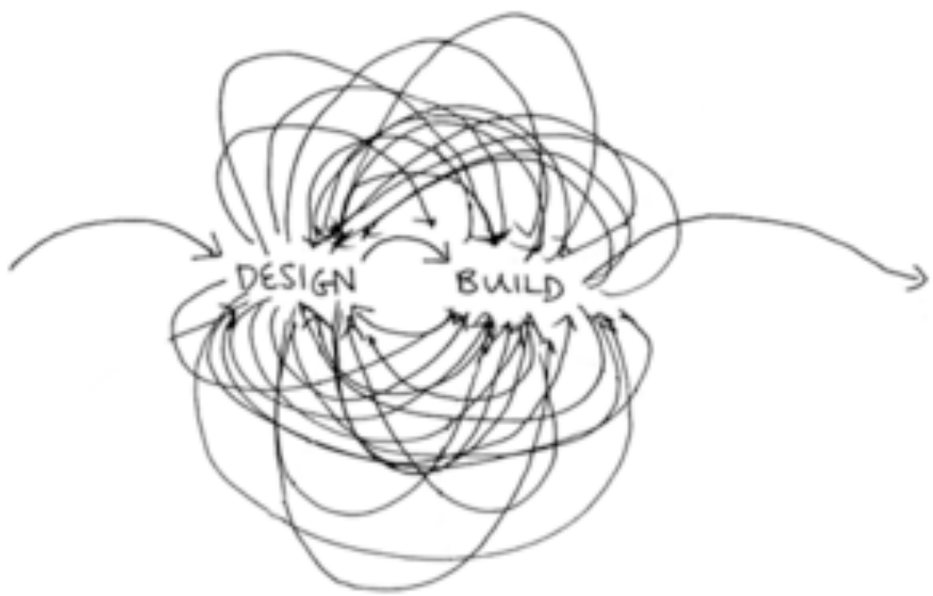
Dr. Aonghus Lawlor

aonghus.lawlor@insight-centre.org

BACKGROUND

Software Engineering Process

customer requirement/concept



software application

- software process
- systematic
- formal methods

Your Task:

When presented with any bus route, an origin stop and a destination stop, a time, a day of the week, current weather, the system should produce and display via the interface an accurate estimate of travel time for the selected journey.

Your Task:

When presented with any bus route, an origin stop and a destination stop, a time, a day of the week, current weather, the system should produce and display via the interface an accurate estimate of travel time for the selected journey.

SOFTWARE PROCESS

Process models

an iterative process repeats one or more elements before moving to the next

System Design
Once the requirements are finalised, a blueprint of the system is drawn to facilitate the implementation process

Requirement Specifications
The requirements for the software, in terms of both design and functionality is captured

Design Implementation
The blueprint of the design is verified and the implementation of the system begins according to the blueprint.

Software Maintenance
Updating, verification and debugging of software

Verification and Test
The implemented system is verified and tested by the team. Once testing is completed, further changes in the requirements are not possible

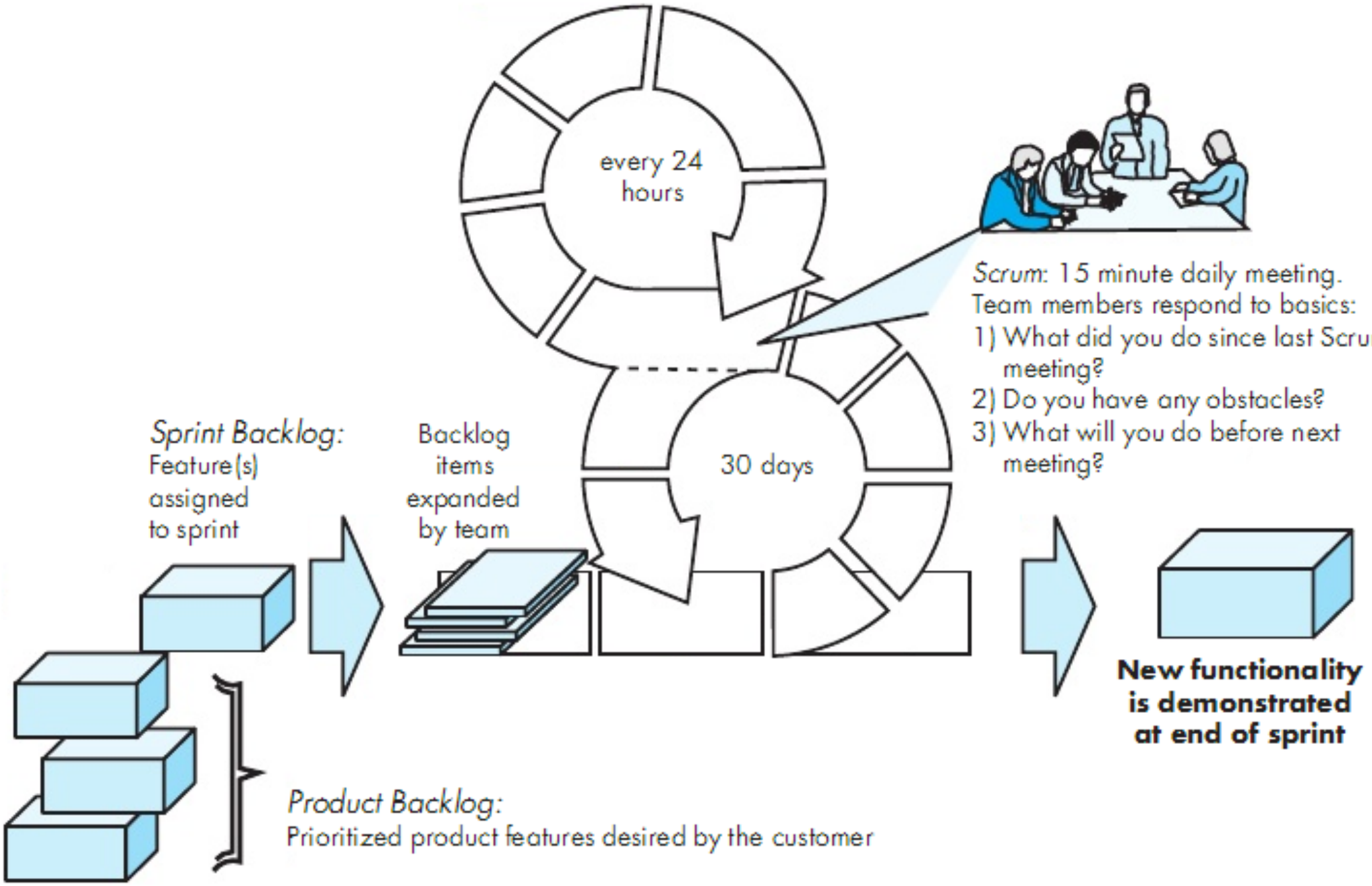
System Deployment
Setting up the system after a pilot or prototype is complete.

Agile Manifesto

Individuals and interactions	over	Process and tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan

http://www.agilemanifesto.org/

Scrum

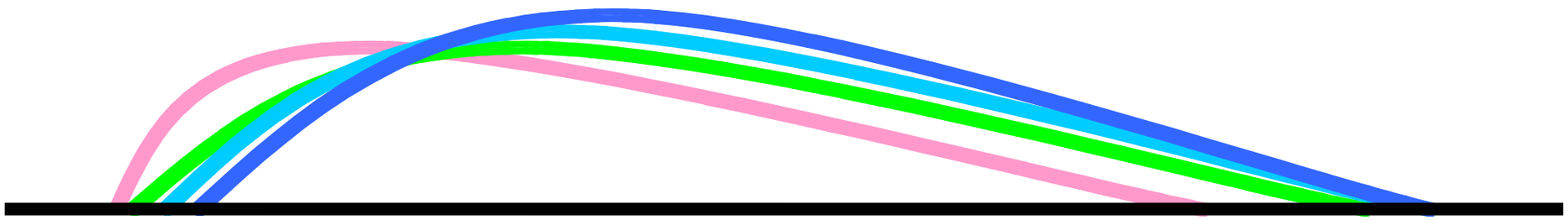

every 24 hours
30 days
Scrum: 15 minute daily meeting.
Team members respond to basics:
1) What did you do since last Scrum meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

Sequential vs. Overlapping Development

Requirements | Design | Code | Test

Rather than doing all of one thing at a time...

...Scrum teams do a little of everything all the time



Source: "The New New Product Development Game" by Takeuchi and Nonaka. *Harvard Business Review,* January 1986.

TEAM FAIRY DUST

AVATARS

-NPI/FEATURES
-CRD
-DOCUMENTS
-IRD

SCRUM MASTER
NITIN

TARGET VELOCITY - '30' STORY POINTS

"CRD/IRD"
(UNDERLINED IN RED
COMMITTED FOR SPRINT)

PRODUCT BACKLOG

GRAHAM

Backlog		W.I.P			
			Tasks →		
Unprepared 10	For Kick Off 6	Story 5	To Do	In Progress	Out
		US3203 Create Geo-Schematic Layout (5) US3204 Integrating extractor (1) US3205 Integrating Geo-schematic common layout (1)		REVIEW	
		US3206 Layout Properties Dialog (8)	REVIEW		
		US3254 Schematics image builder.			
		US3248 PNI Make Geo Schematics (again) (4)			
		US2529 Core customers can create geo-Schematics TEST (8)			

Epics/ Placeholders

Stories Sized Understood

Kicked Off Task Created

CODE | TEST


BRONX

Performance

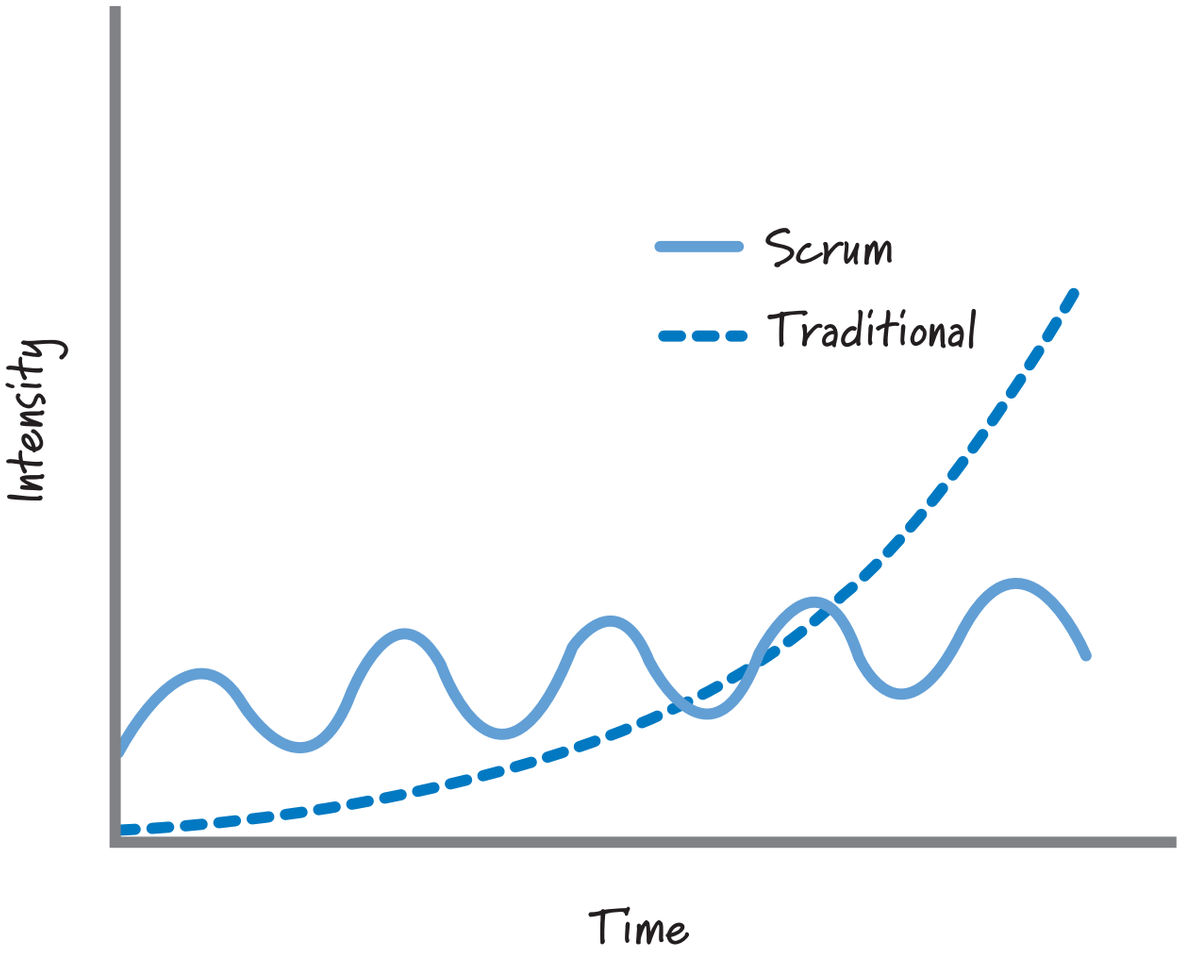
within a sprint we'll likely see intensity increase a bit near the end of the sprint

the overall intensity of work during each sprint should closely resemble the intensity of the previous sprint, reinforcing the team's working at a sustainable pace.

levelling the work

it doesn't come in huge chunks or intense bursts, especially late when it is most harmful.

will likely work fewer overtime hours



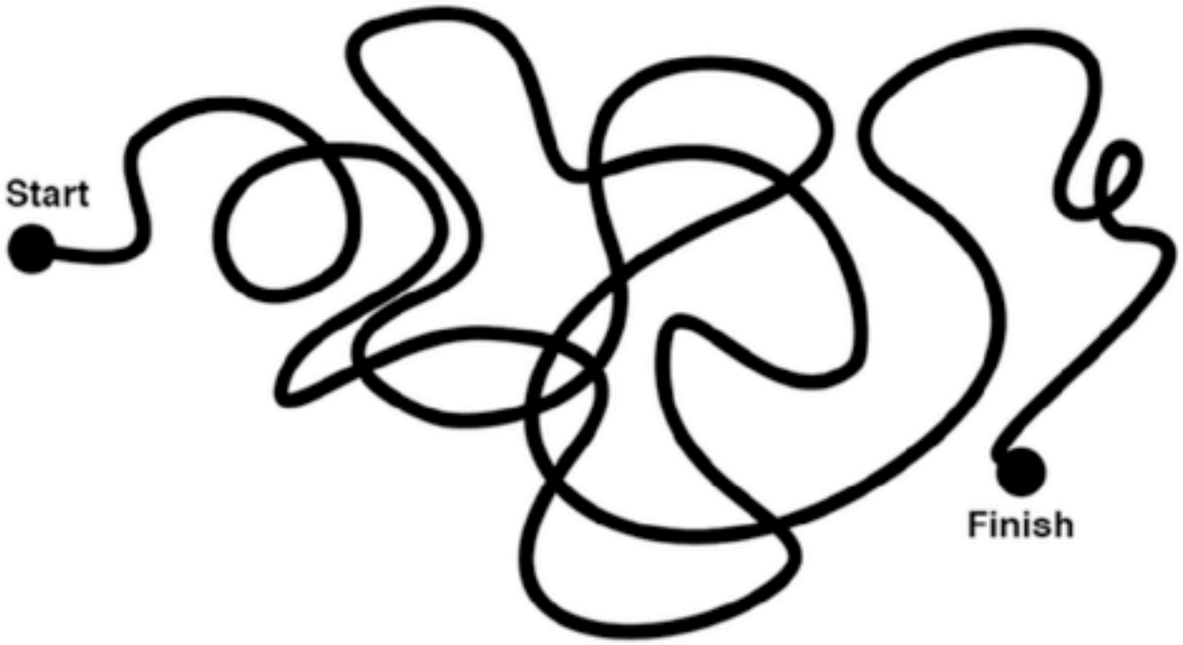
SOFTWARE DESIGN

"That's been one of my mantras - focus and simplicity. Simple can be harder than complex: You have to work hard to get your thinking clean to make it simple."

"That's been one of my mantras - focus and simplicity. Simple can be harder than complex: You have to work hard to get your thinking clean to make it simple."

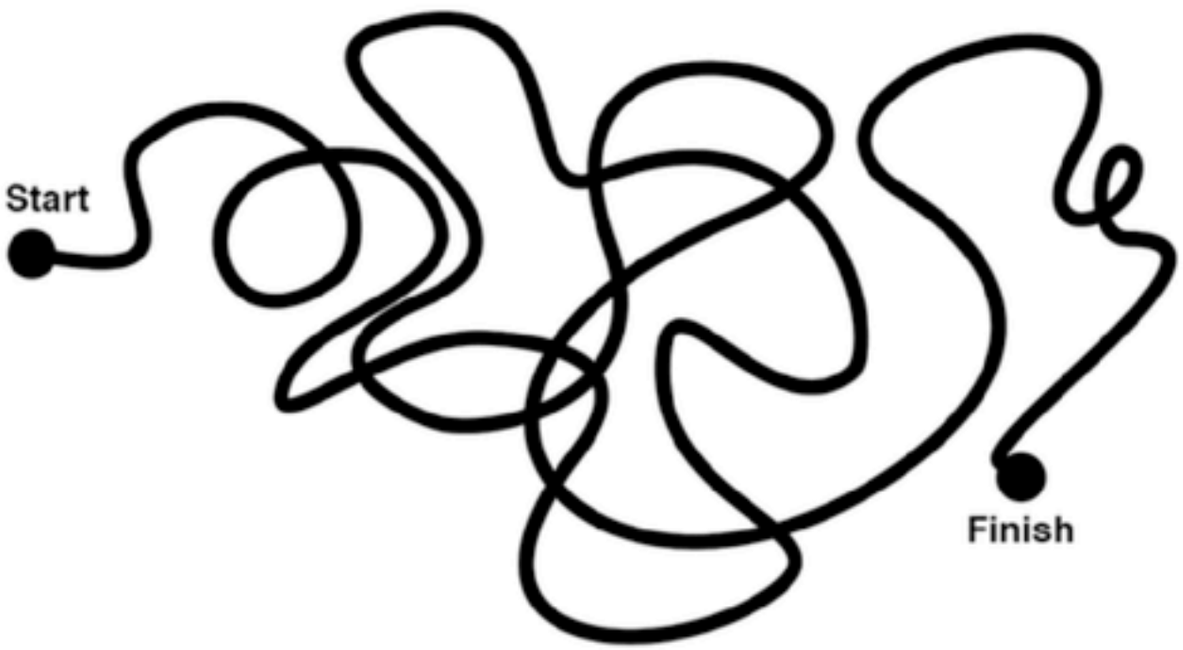
Steve Jobs

Experience

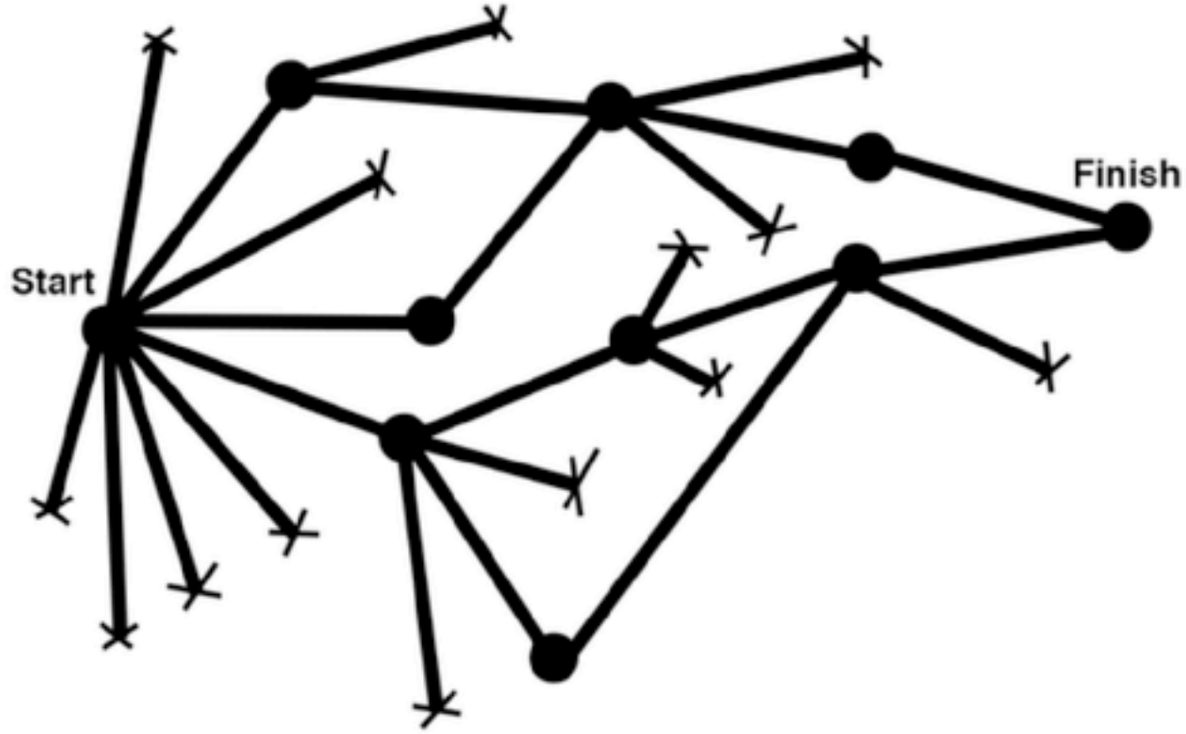


How a Junior Solves Problem

Experience



How a Junior Solves Problem

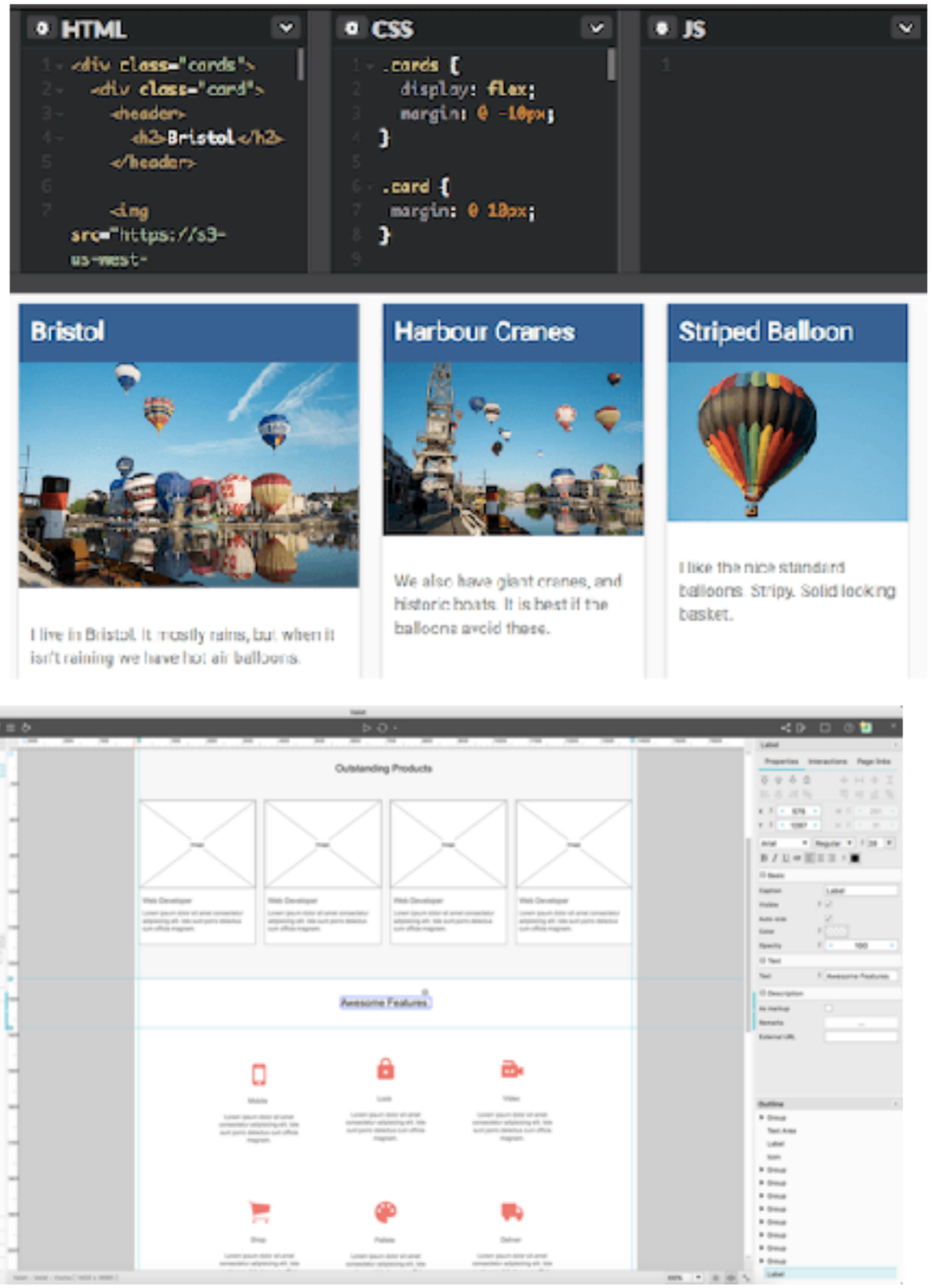


How a Senior Solves Problem

Prototypes

You can prototype

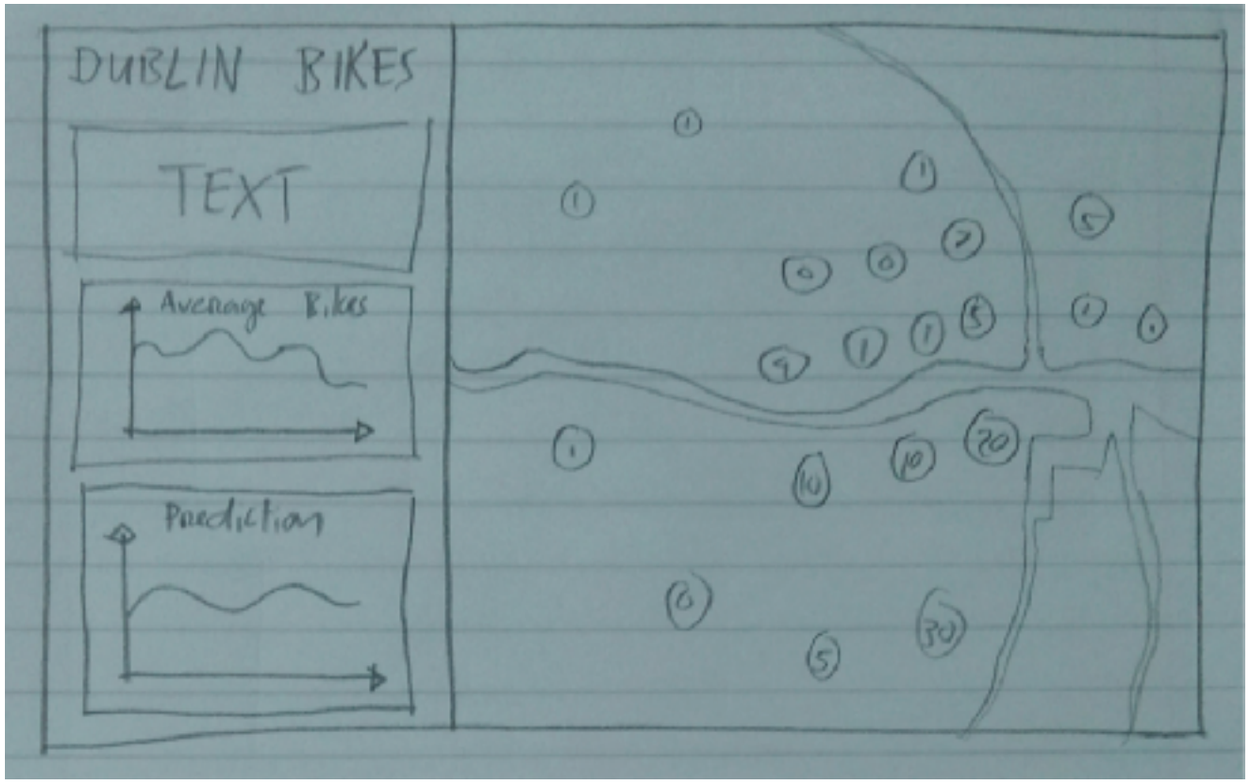
- Architecture
- New functionality in an existing system
- Structure or contents of external data
- Third-party tools or components
- Performance issues
- User interface design

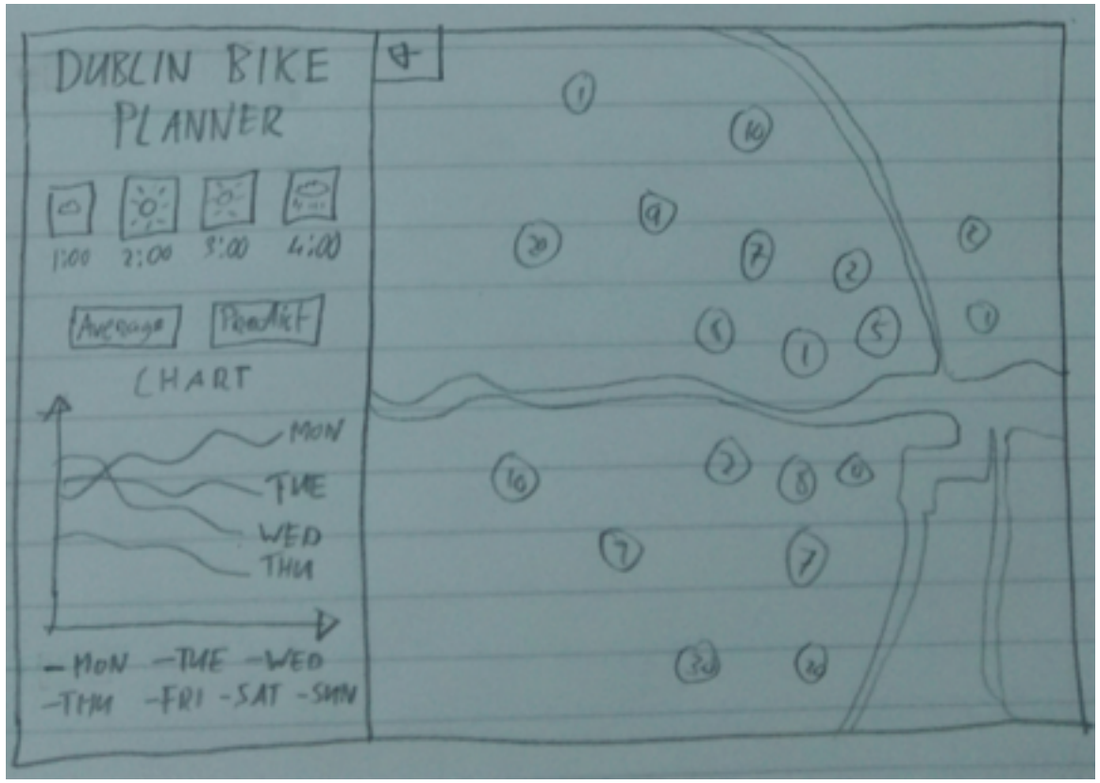


“The sooner you start to code the longer the program will take.”

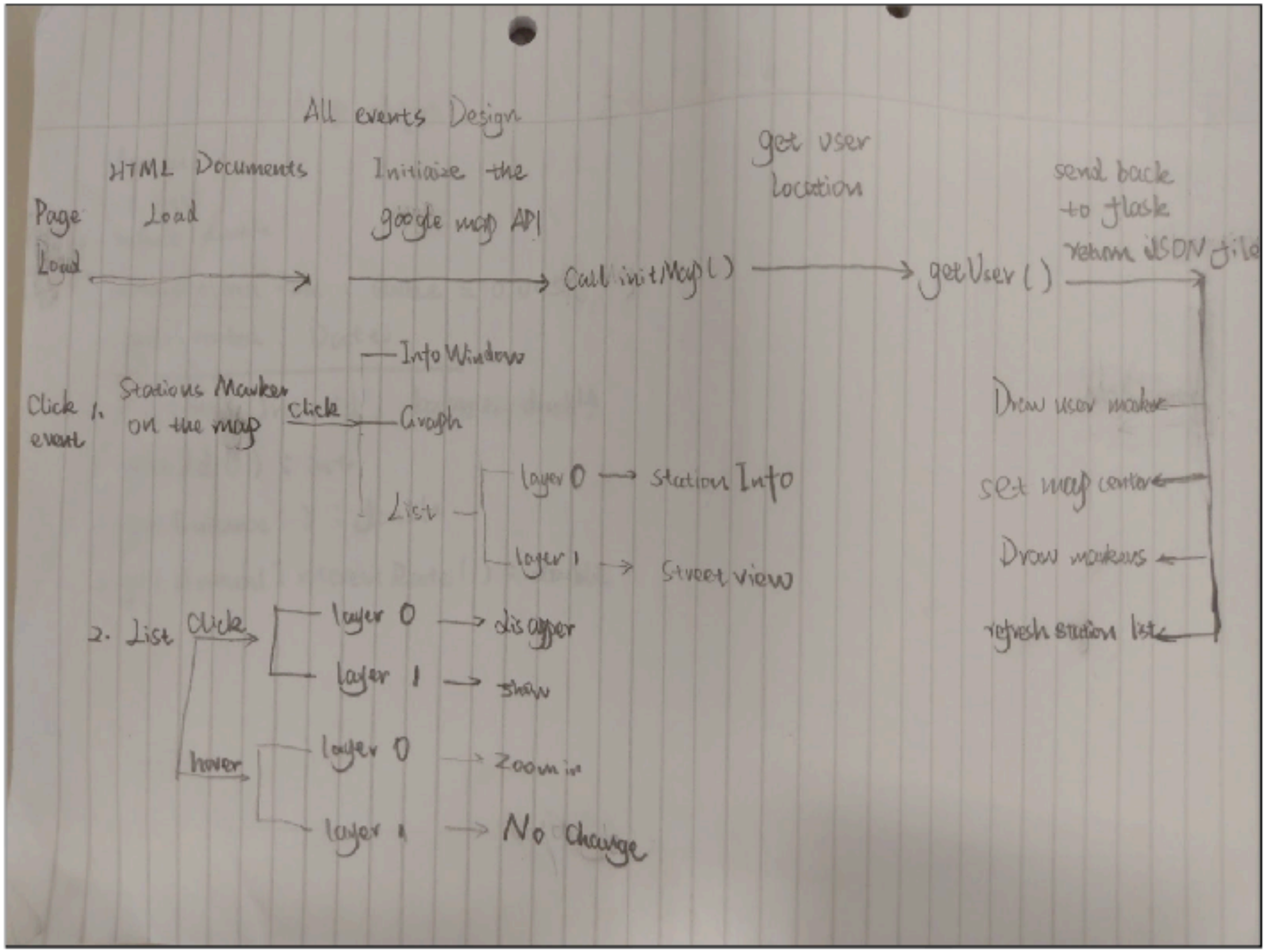
Roy Carlson

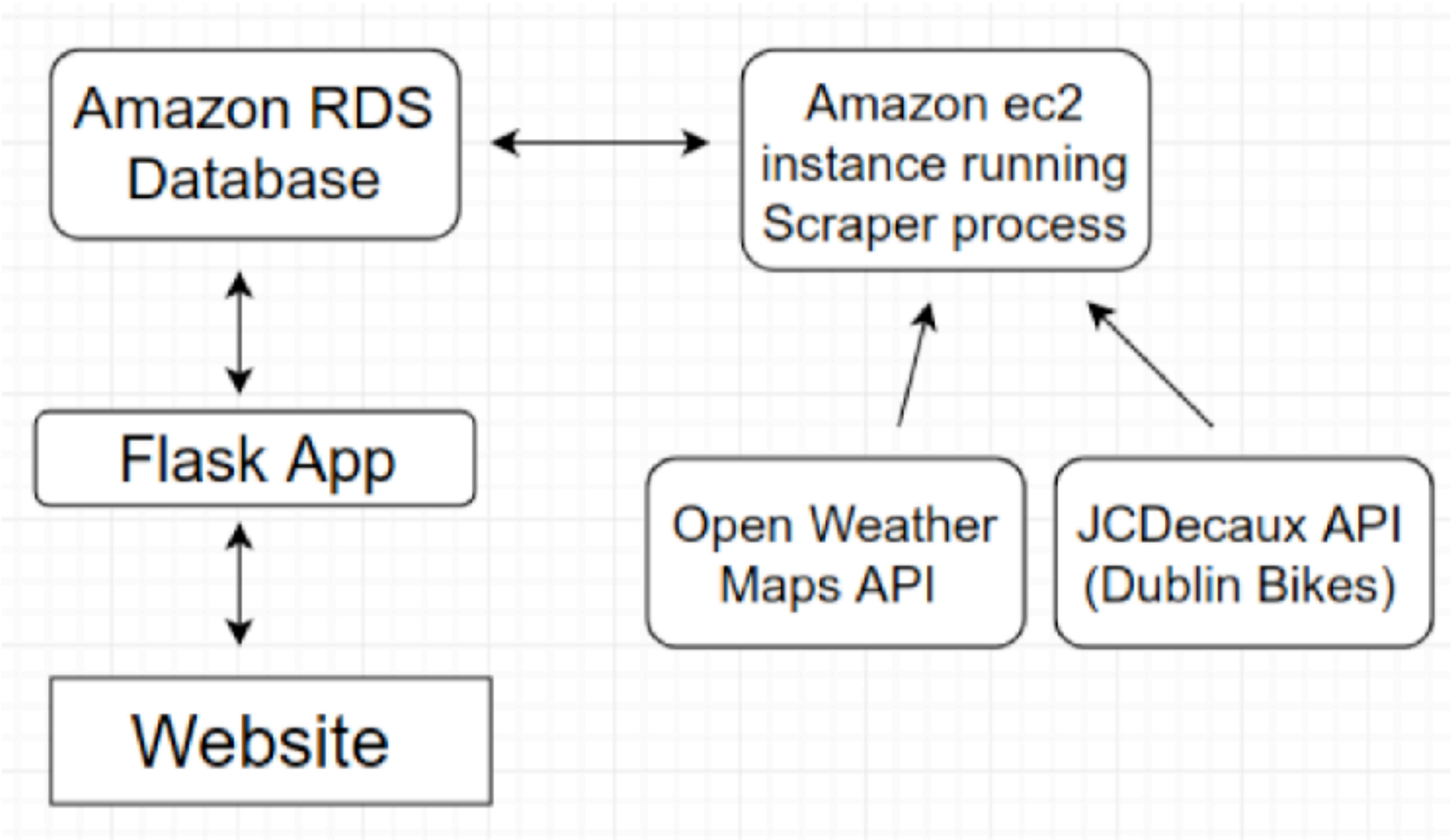
University of Wisconsin


DUBLIN BIKES
TEXT
Average Bikes
Prediction


DUBLIN BIKE PLANNER
1:00
2:00
3:00
4:00
Average
Predict
CHART
MON
TUE
WED
THU
MON
TUE
WED
THU
FRI
SAT
SUN

Sketched the interactive design of the webpage




Amazon RDS Database
Amazon ec2 instance running Scraper process
Flask App
Open Weather Maps API
JCDecaux API (Dublin Bikes)
Website

A Web Page

http://dublinbikes.com

Plan your Jurney

Choose a date:

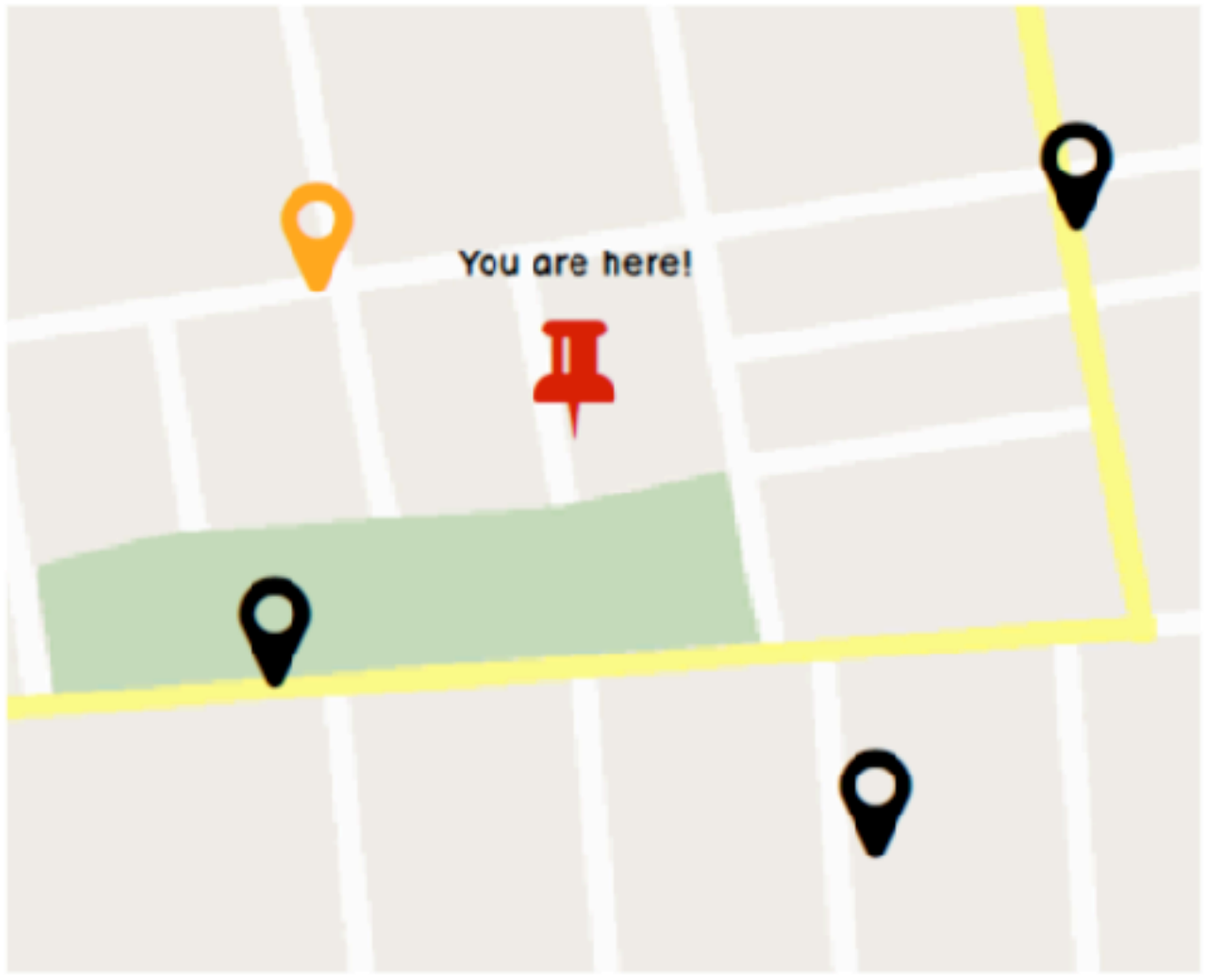
FEBRUARY 2019

S	M	T	W	T	F	S
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	1	2
3	4	5	6	7	8	9

Choose a time:

OK

Select a station near you

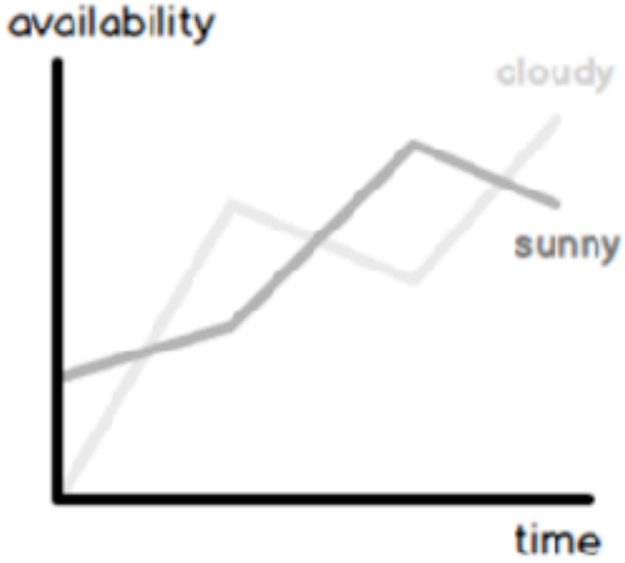


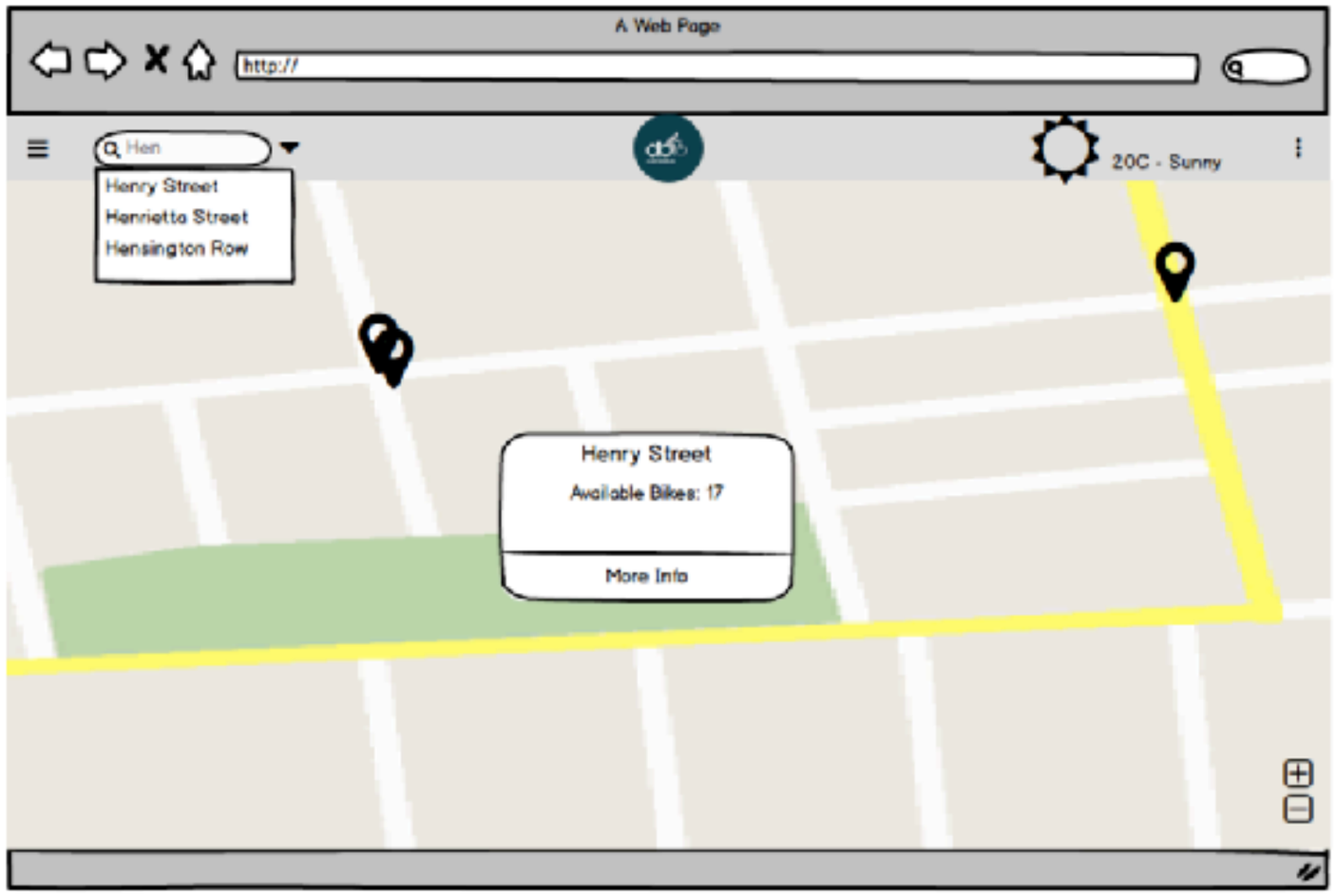
Selected Station

Station_Name
Address
Forecasted availability

Weather forecast

now later




A Web Page
http://
Hen
Henry Street
Henrietta Street
Hensington Row
20C - Sunny
Henry Street
Available Bikes: 17
More Info


A Web Page
http://
search
20C - Sunny
Back to Map
Bike stats per day
Stand stats per day
Availability Probability
Bike stats per day
Stand stats per day
Availability Probability

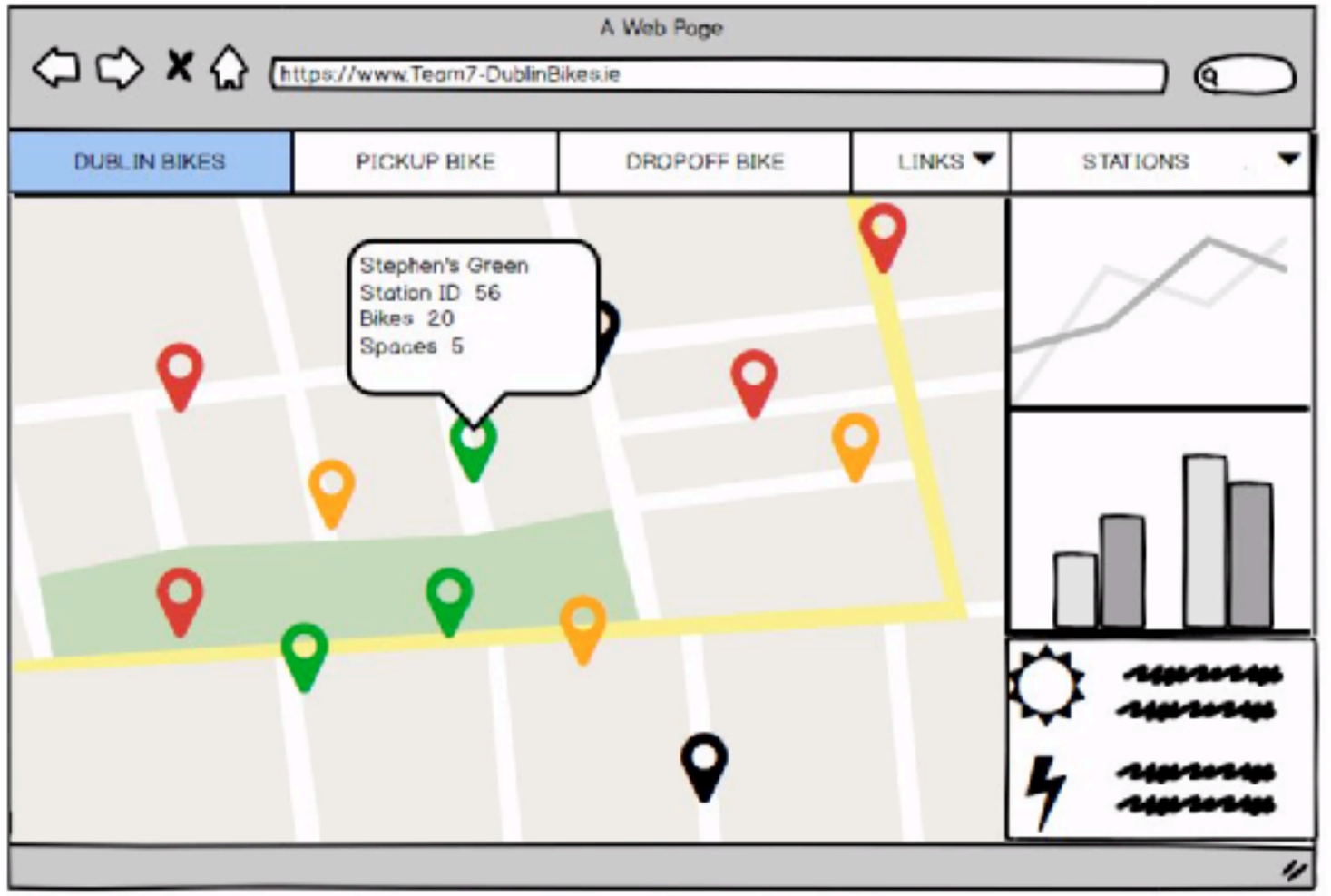


Figure 2: Application mockup created from initial design discussions and sketch

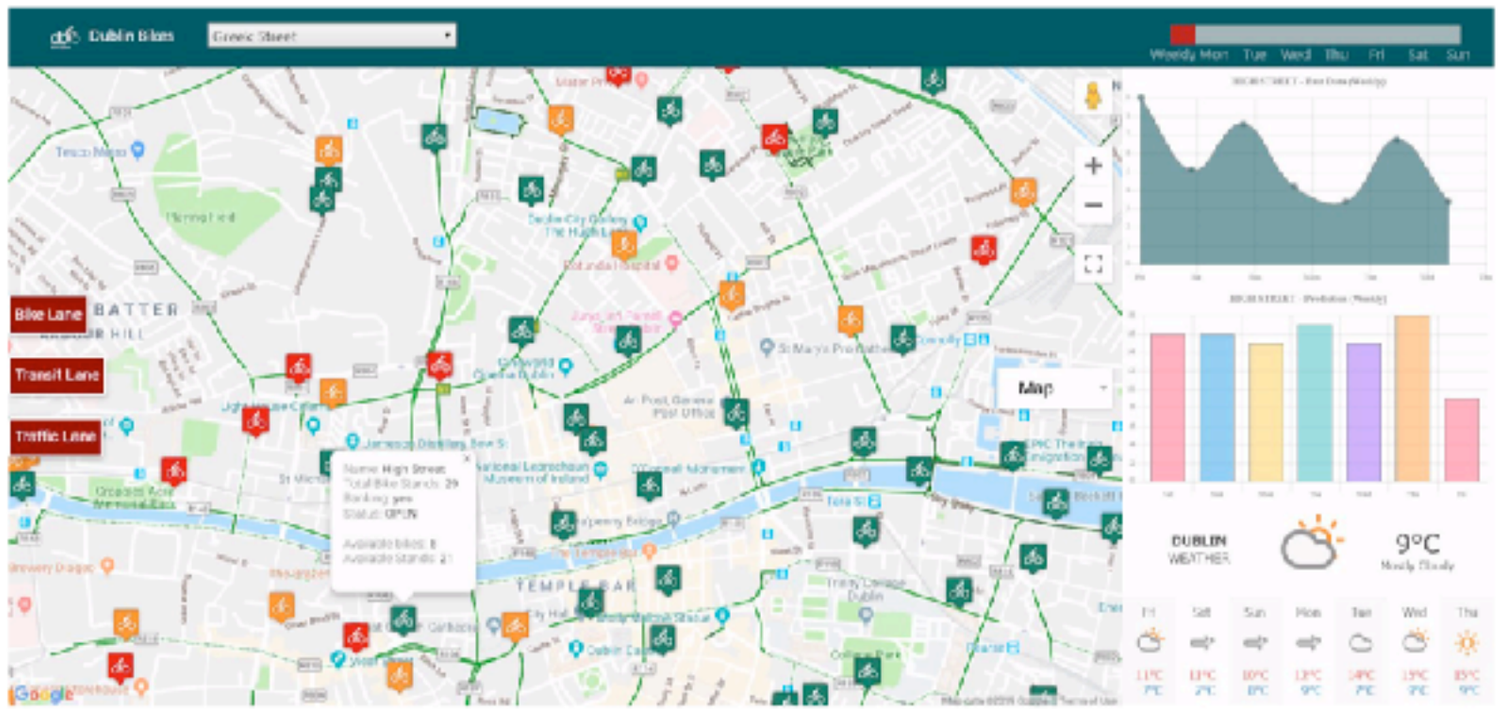


Figure 3: Final webpage for reference

"Good judgement comes from experience, and experience comes from bad judgment"

Fred Brooks

University of North Carolina

Prototypes

When creating a wireframe, the main goal is to convey how a given page will accomplish its intended goal.
polished look-and-feel, or final aesthetics are not important at this stage

Prototypes

When creating a wireframe, the main goal is to convey how a given page will accomplish its intended goal.
polished look-and-feel, or final aesthetics are not important at this stage

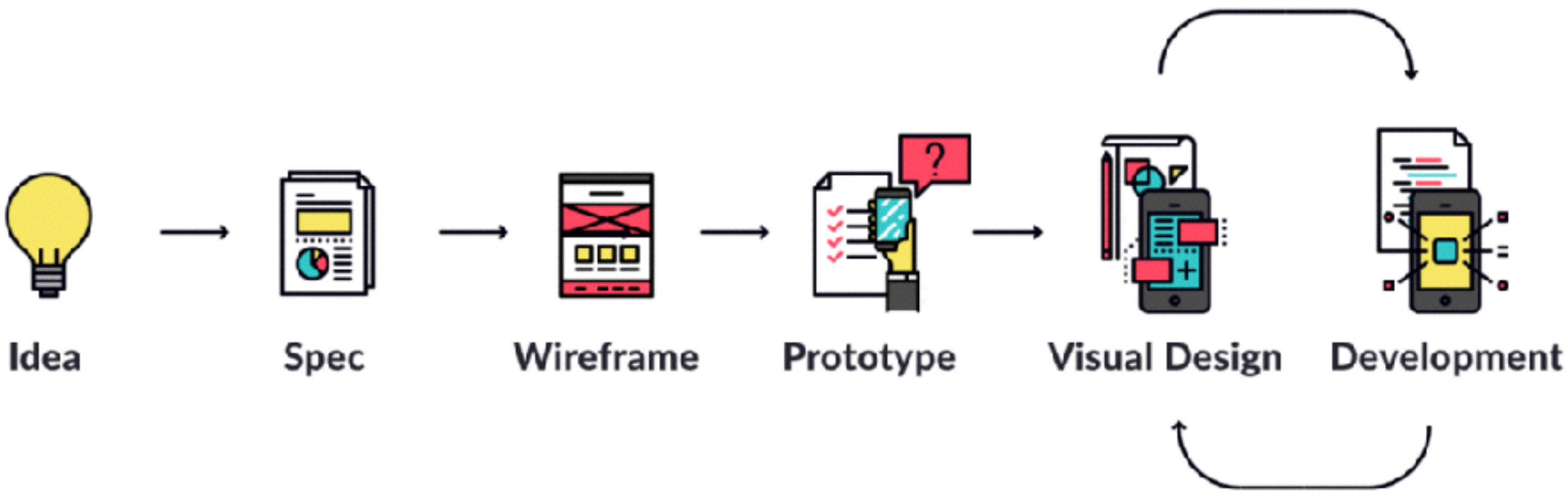
Prototypes

When creating a wireframe, the main goal is to convey how a given page will accomplish its intended goal.
polished look-and-feel, or final aesthetics are not important at this stage

general ideas -> main objectives -> wireframes -> coding the pages with their core functionality -> implementing the final design

Prototypes

When creating a wireframe, the main goal is to convey how a given page will accomplish its intended goal.
polished look-and-feel, or final aesthetics are not important at this stage



"Prototyping cuts the work to produce a system by 40%."

Larry Bernstein
Bell Labs

Prototypes

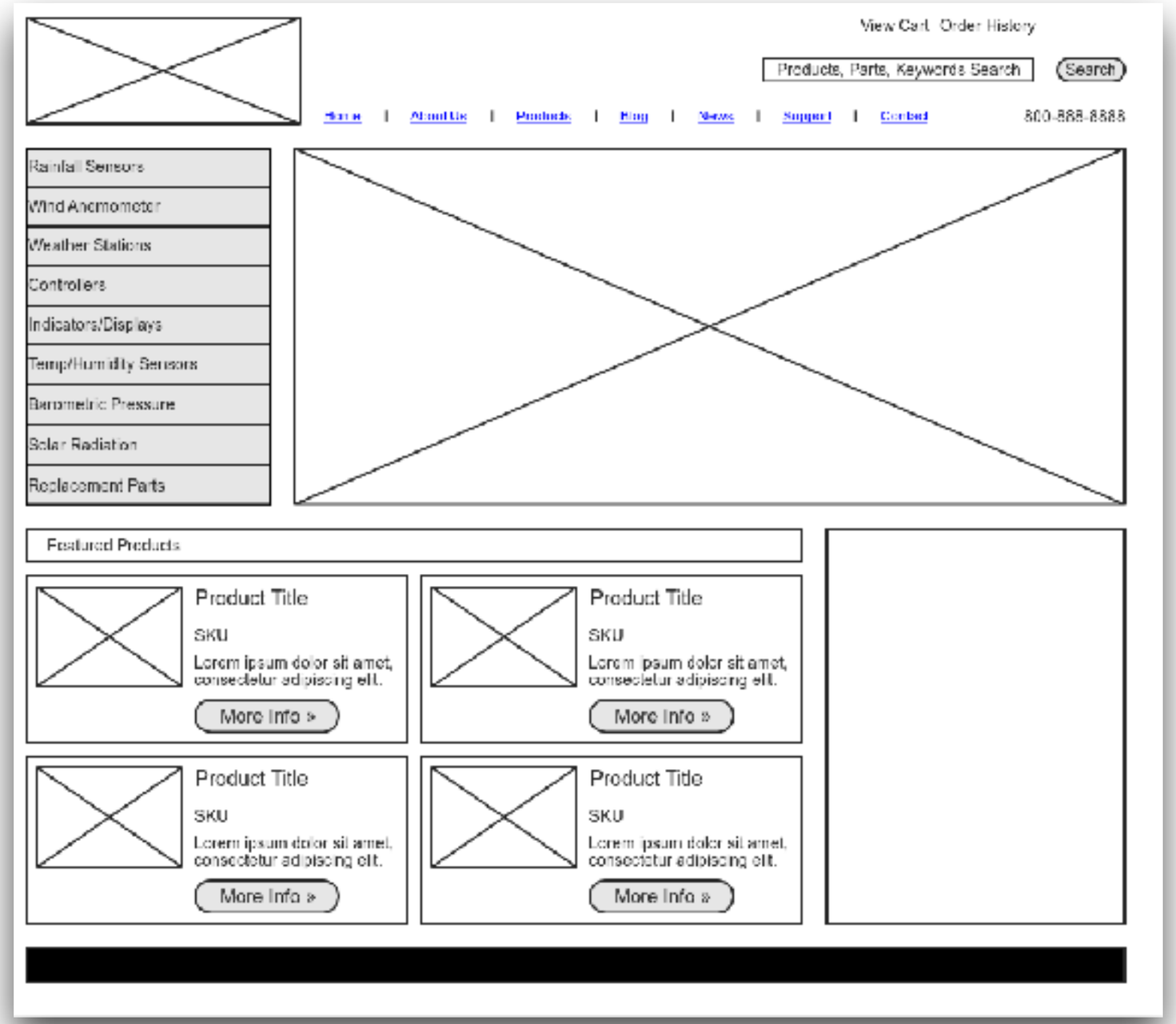
- **Is this idea viable?**
- **Is this idea technically feasible?**
  Can this be made? Who should make it? How would we go about making it? What sort of tools do we use? What sort of data/API/touch points do we need? What are the obstacles facing the implementation of the idea?
- **Is someone else already doing this?**
  Most things today are a remix. While that's cool, what we can do better? What components of this idea differentiate it from existing ideas?
- **Could this be made simpler/ differently?**
  Are there other ways to accomplish the same goals? Or, are there other methods that could be more effective and take less time to execute?

Prototypes

- Get an idea
- Write it down
- Build a prototype
- Iterate on design and development until you have something good.

Prototypes

- Get an idea
- Write it down
- Build a prototype
- Iterate on design and development until you have something good.



Web Frameworks

- A framework comes up with predefined components, so you can make use them directly rather than development from scratch.

- No more tweaking of the base to make it look same across all the browsers. The **CSS framework** will do that.

- No more adjusting of pixel widths and wondering if the sidebar has standards compliant for widgets or images.

- Synergy is maintained across the website and on all web pages because the framework will remember everything that you did on each web page saving you the time and energy to try and remember specifications of each element.

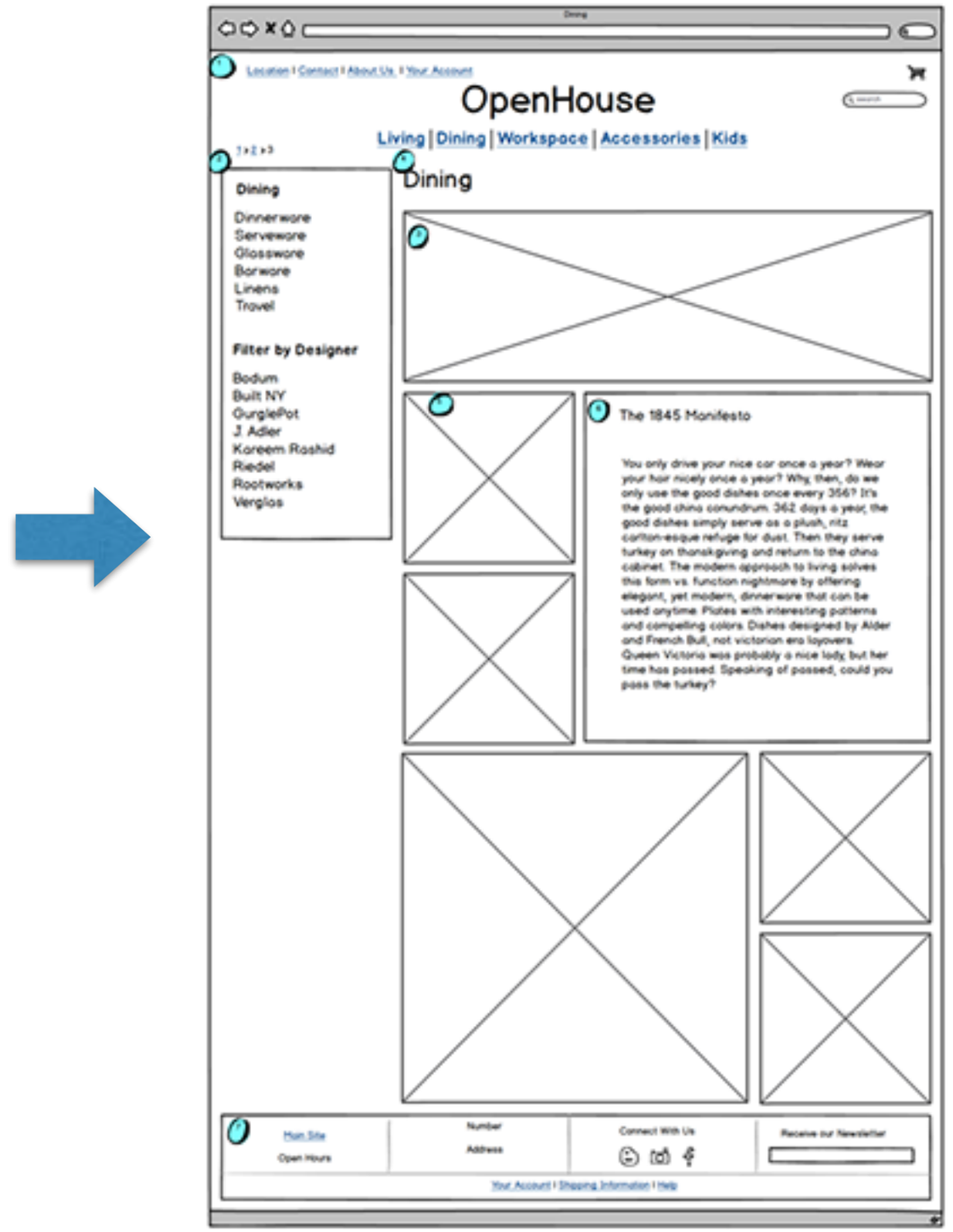
1. Fewer Cross browser bugs
2. A consistent framework that supports major of all browsers and CSS compatibility fixes
3. Lightweight and customisable
4. Responsive structures and styles
5. Several JavaScript plugins using the jQuery
6. Good documentation and community support
7. Loads of free and professional templates, WordPress themes and plugins
8. Great grid system

Web Frameworks

- Since a framework has a standard set of grids and selectors and other codes, it restricts your design. So for every new kind of design idea you would want to implement, you would have to learn a new framework.

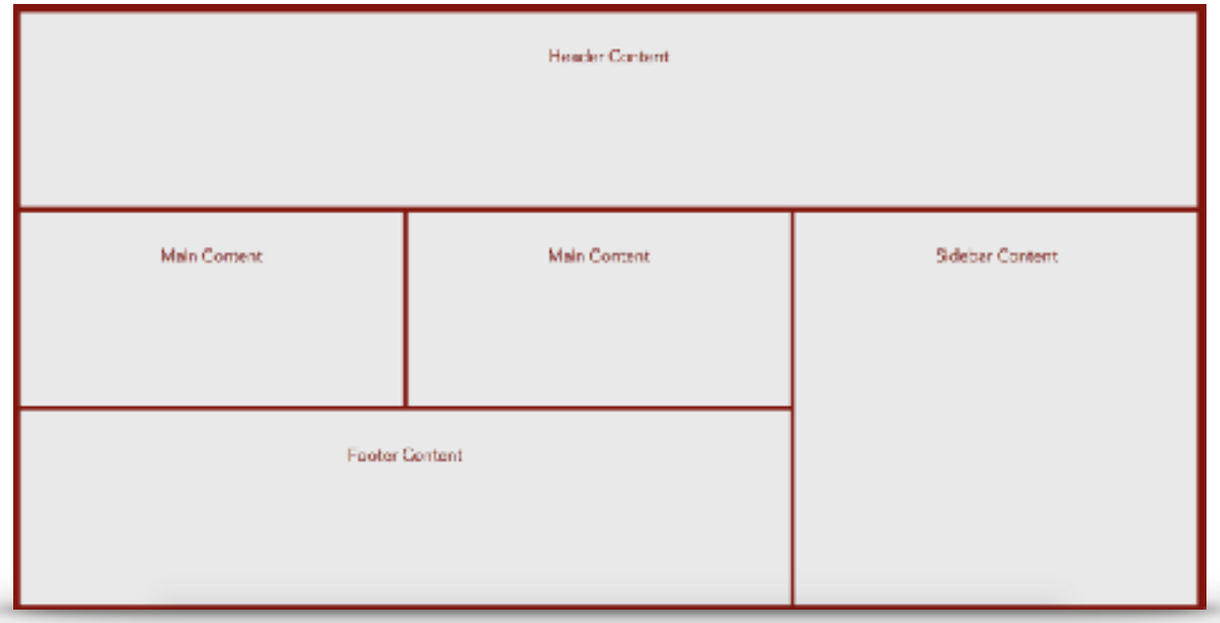
1. Deviations from standard designs can cause a lot of work
2. Customisation is required to distinguish your design- can be extra work.

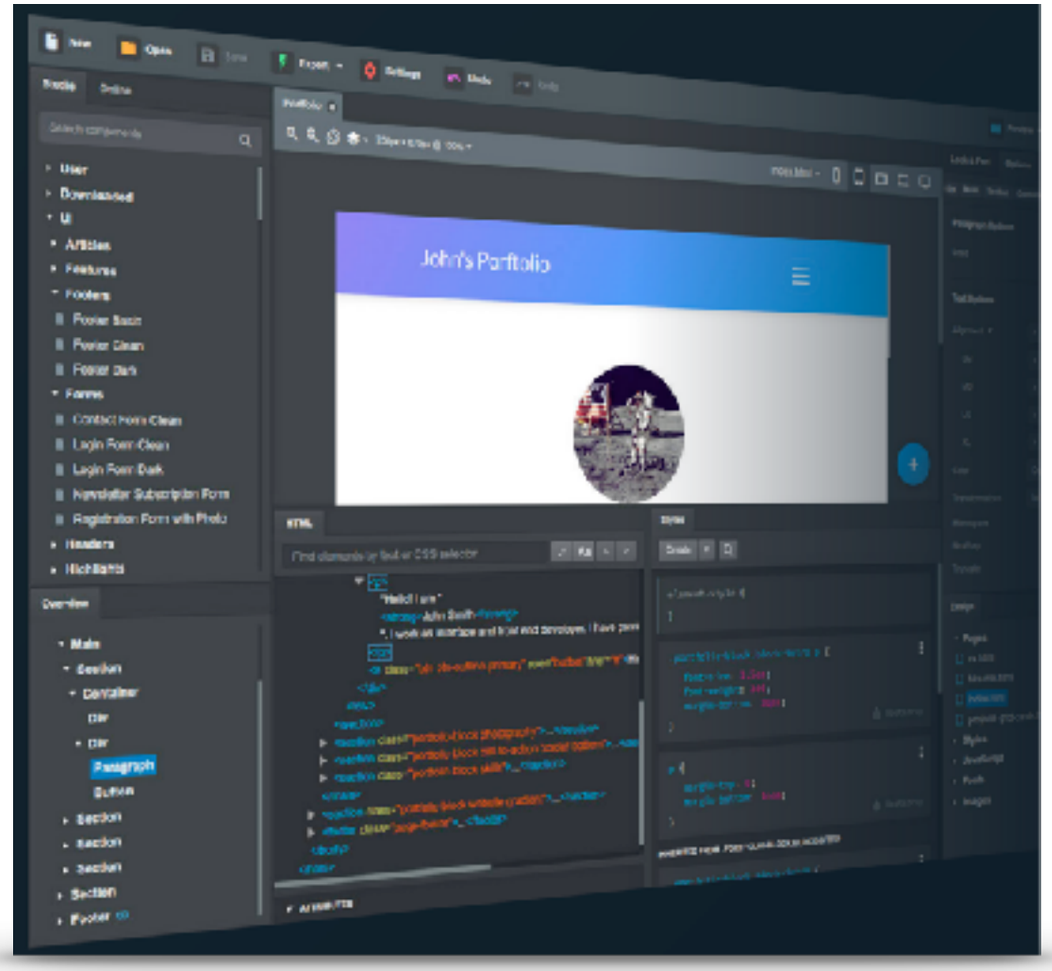
Components

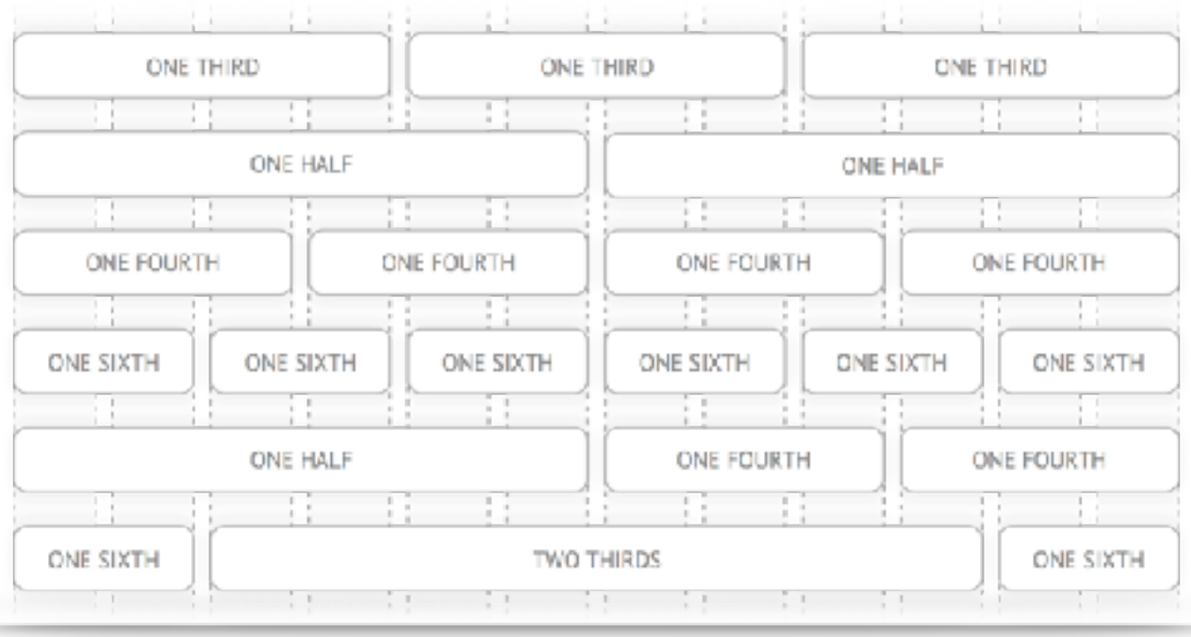


Use the standard bootstrap components to compose your main site. The components can be easily customised and replaced or upgraded.

Layout tools



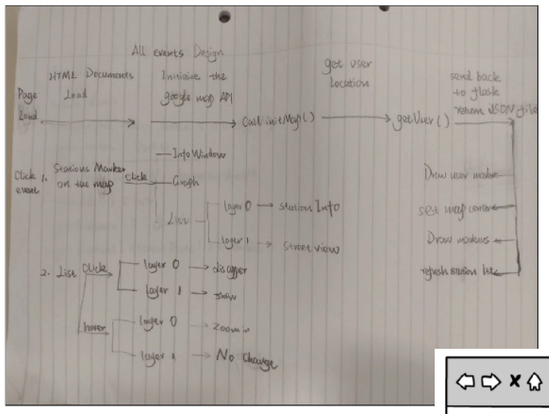


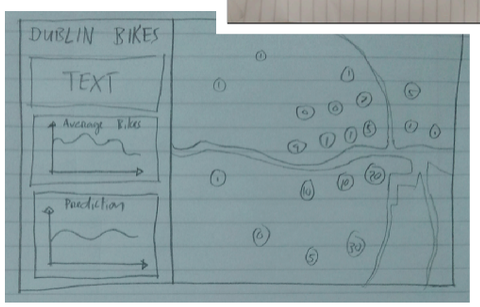


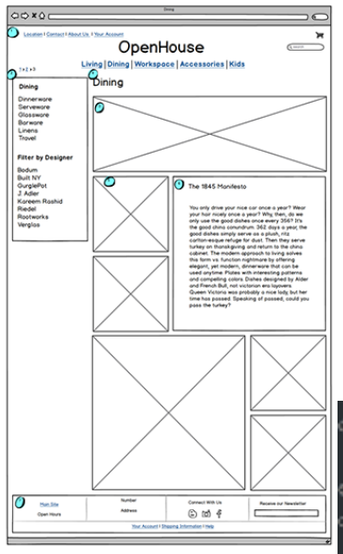
https://www.sitepoint.com/bootstrap-tools-playgrounds/
https://medium.com/codingthesmartway-com-blog/introduction-to-bootstrap-4-flex-layout-flexbox-for-bootstrap-f85405ce5ebf

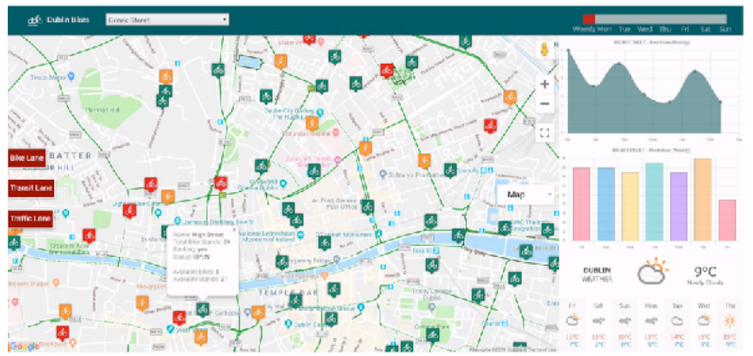
Prototyping

Sketched the interactive design of the webpage

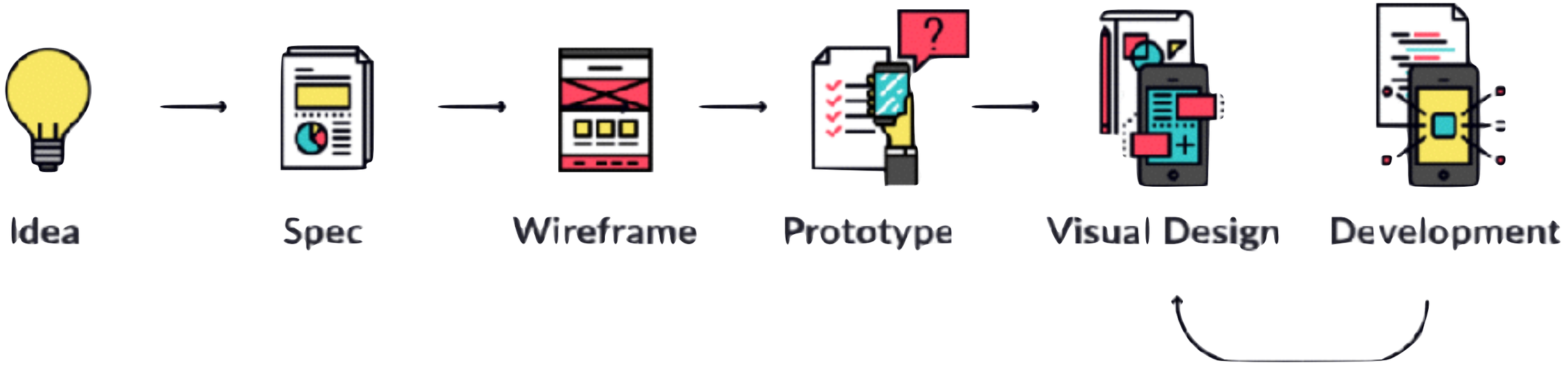








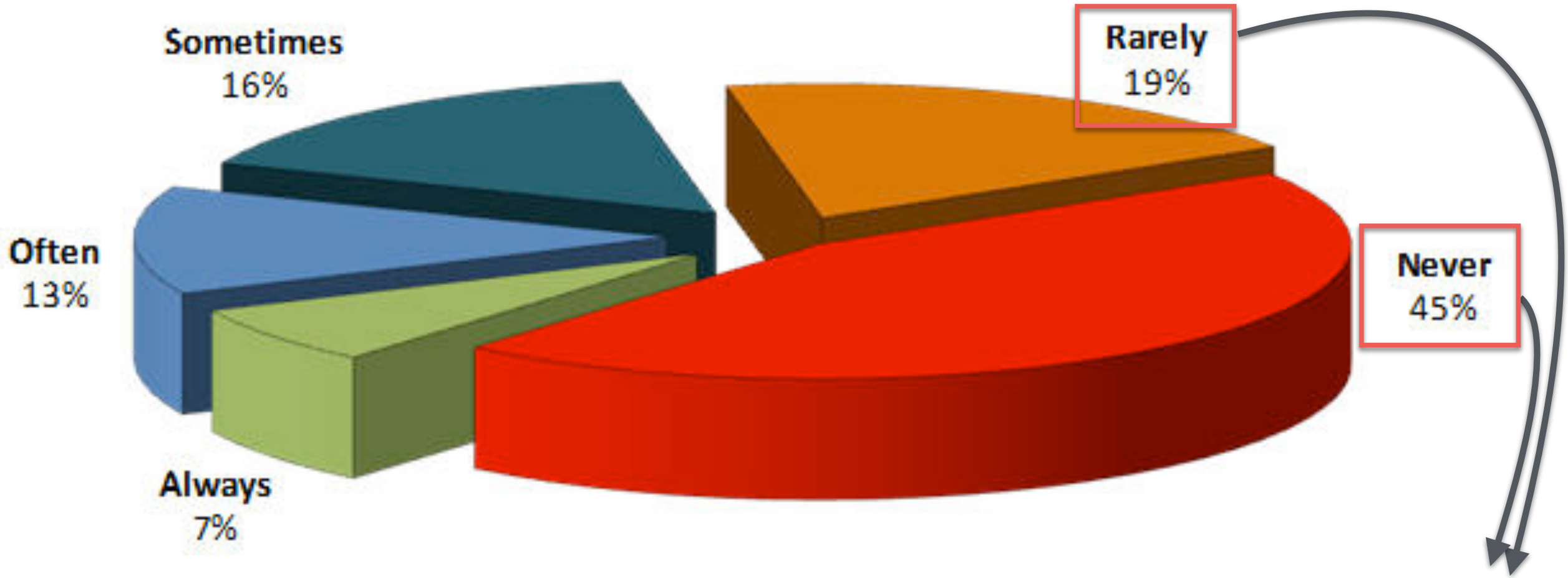

```
<div class="row">
    <div class="col-xs-12 header">HEADER</div>
</div>
<div class="row">
    <div class="col-xs-4 menu">MENU</div>
    <div class="col-xs-8 content">CONTENT</div>
</div>
<div class="row">
    <div class="col-xs-12 footer">FOOTER</div>
</div>
```



Wrong Priorities → Waste of time

Requirements:

Features and functions
used in a typical system



Rarely + Never = 64%

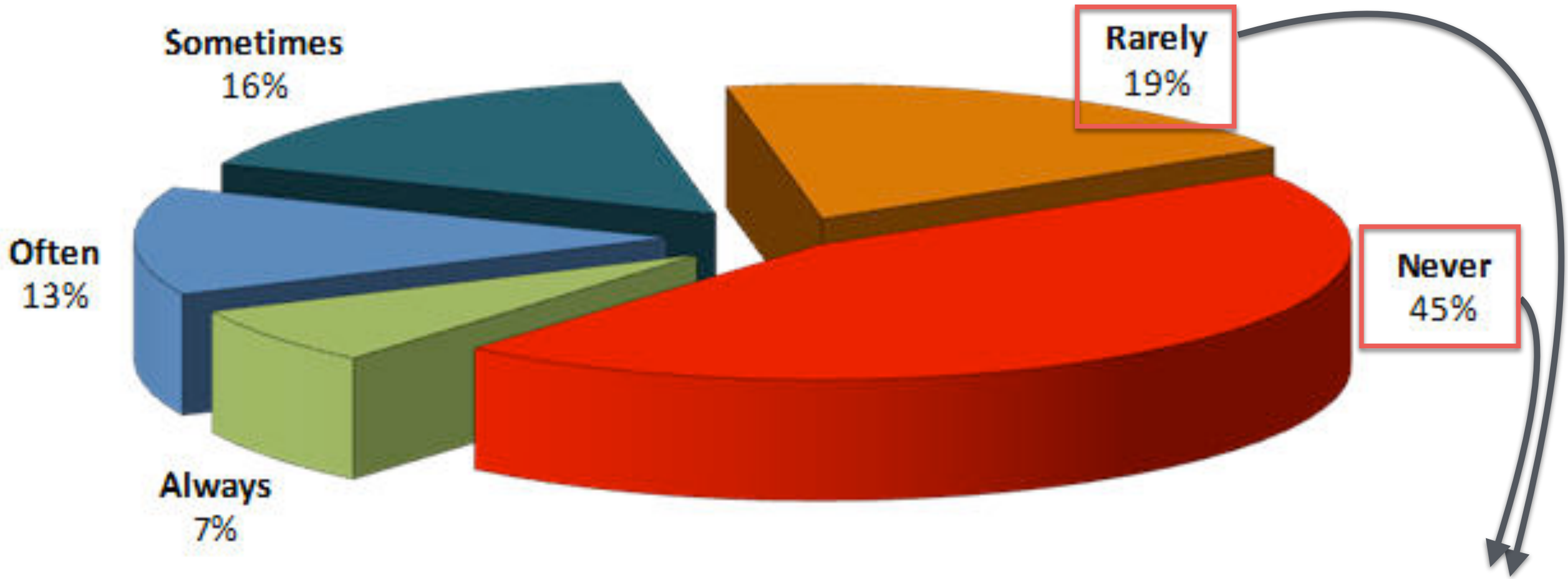
Standish Group Study Reported at XP2002 by Jim Johnson, Chairman

Wrong Priorities → Waste of time

Features and functions used in a typical system

Requirements:

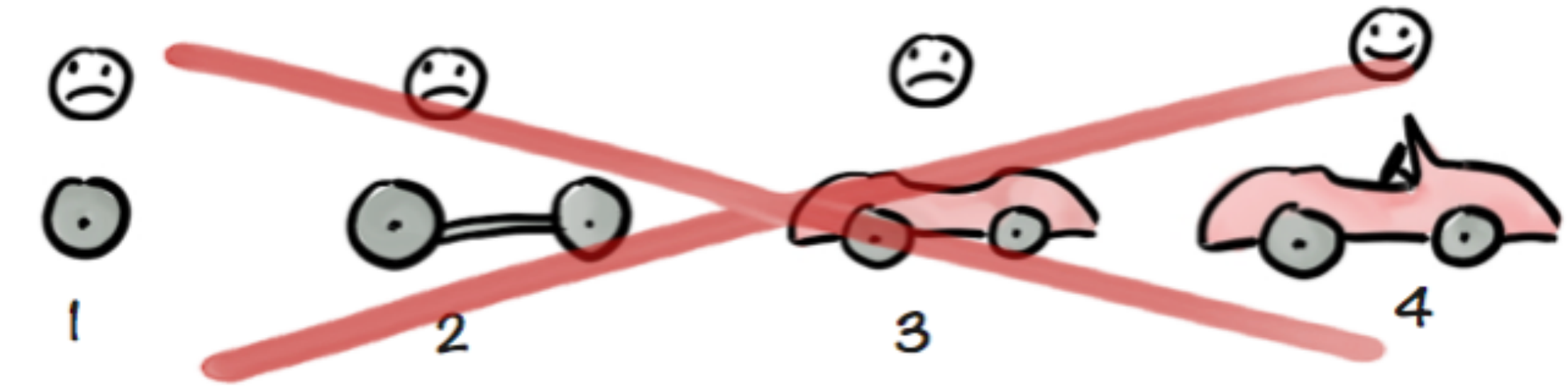
"anything that drives design choices"

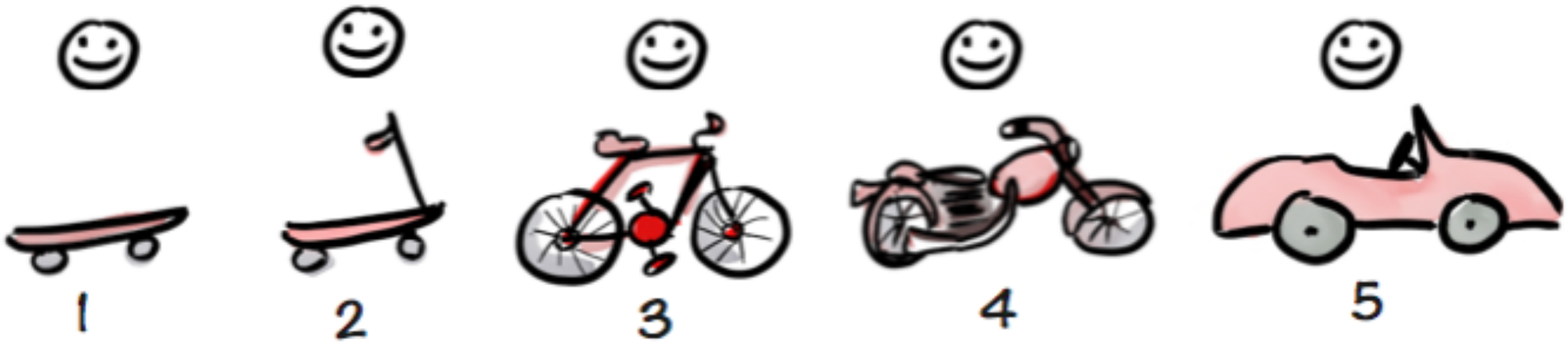


Rarely + Never = 64%

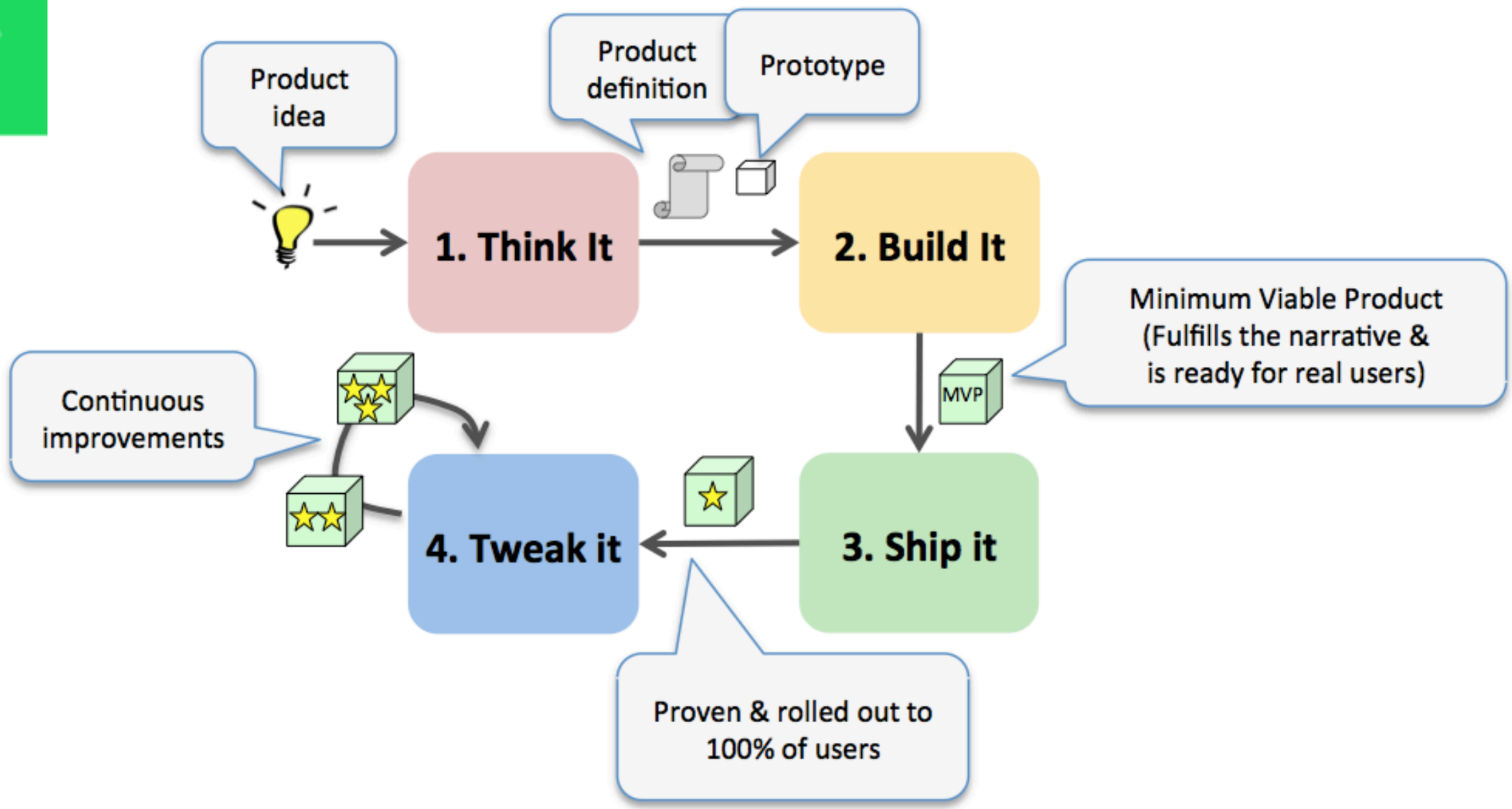
Standish Group Study Reported at XP2002 by Jim Johnson, Chairman

Spotify engineering culture (part 1), (part 2)





- phased approach
- deliver value at every stage by releasing a working product that can be used
- not the final product, but it still usable and does the job
- enhance progressively
- The idea is turn around something tangible as early as possible and concentrate on each stage at a time.



- **Think It** = figure out what type of product we are building and why.
- **Build It** = create a minimum viable product that is ready for real users.
- **Ship It** = gradually roll out to 100% of all users, while measuring and improving.
- **Tweak It** = Continuously improve the product. This is really an end state; the product stays in. **Tweak It until it is shut down or reimagined (= back to Think It).**

Perfectionism

- Does this change *really* make a difference? Is it worth my time? Consider engineering costs vs. value created. Think long- and short-term.

- Does it provide value to the users of my software? Users typically don’t care about internals like program code.

- Does it matter to my coworkers? My boss? My future self?

- More often than not, settling for *good enough* is a valid choice to make shipping working code is highly valued.

- The best code is no code at all. Perfectionist or not, always start by looking for solutions that don’t involve writing code.

Great software today is often preferable to perfect software tomorrow.

Tools


eclipse


pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$


Trello


MySQL


python


JavaScript


HTML


Basecamp®


GitHub


amazon
web services™

Magpie Syndrome

- Did you ever make a choice of a new tool before you fully understood it?
- We seem to crave shiny new things even if it has little value.
- This is called Magpie Syndrome.
- Every technical decision will have a business, development, or operational impact as time goes on, and we should keep this in mind.

The Marketing Behind MongoDB

Countless NoSQL databases competed to be the database of choice. MongoDB's marketing strategy helped it become the winner.

ADVICE

Benefits of Agile

Incorrect approaches are quickly identified	a wrong path is much harder to correct the later it is identified. Agile emphases "failing fast" by showing the business progress every day.
Decisions are made quickly	when questions arise, it is common to hold a quick meeting of the team to discuss the issue
Collaboration results in many benefits	when issues surface, everyone in the group knows about them, and solutions are often identified by resources working on a completely different area of the application.
Change is recognised as inevitable and is embraced	It is understood that no one can define exactly how a system should work at the start of a project- the system is developed iteratively and course corrections are made along the way.
The final product contains the most useful features	identify the features that add value
The technical documentation takes less time and is correct	Traditional development approaches spend a great deal of time on documentation that is often not maintained or used. The milestones in waterfall development are often the creation of documentation rather than actual working code. Agile reduces this load, but there is documentation!

Common Scrum Problems

Fear
often under commit or pad estimates, lack of confidence, experience.
be honest with burndown charts, estimation

Reviews
Good for collaboration, enforcing quality

Communication
trouble ahead if you don't talk to each other!
face2face is best

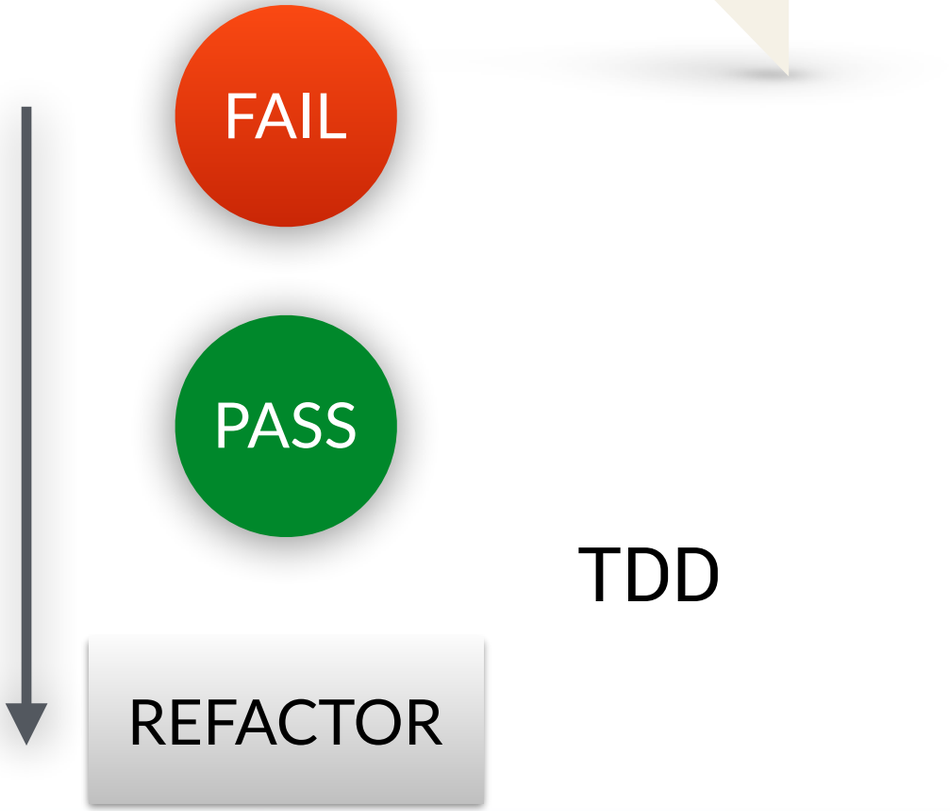
No Retrospectives
Do the retrospective at the end of the sprint and try to fix the issues.
Do a demo and analyse for next sprint

Poor Estimation
Estimating the size and scope of new work can be difficult, especially with a new team.
But- estimation gets easier with time.

Poor Testing
Develop a good testing habit

"If it ain't broke, don't fix it"

Ronald Reagan



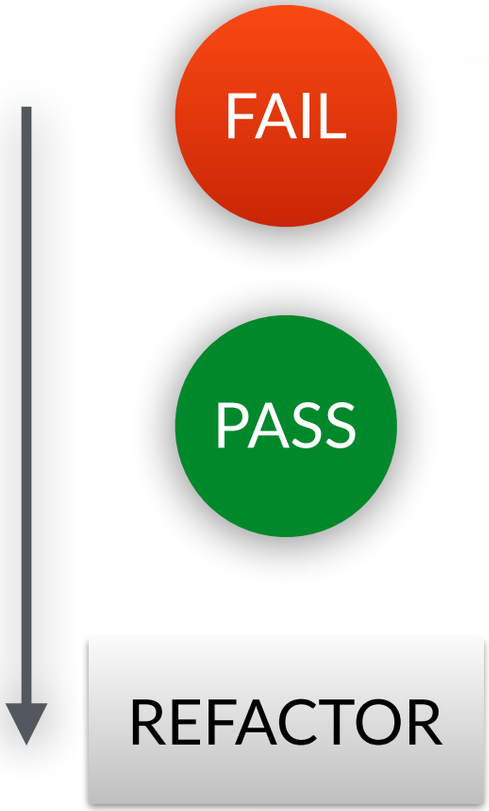
"If it ain't broke, don't fix it"

Ronald Reagan

"If we can't fix it, it ain't broke"

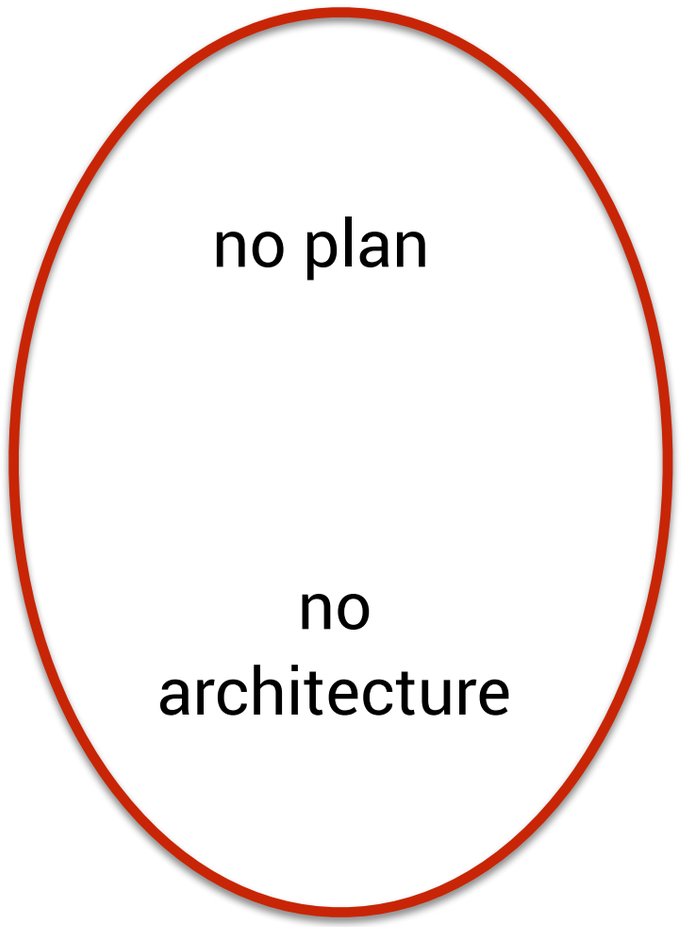
[Maintainers motto]

Lt. Colonel Walt Weir

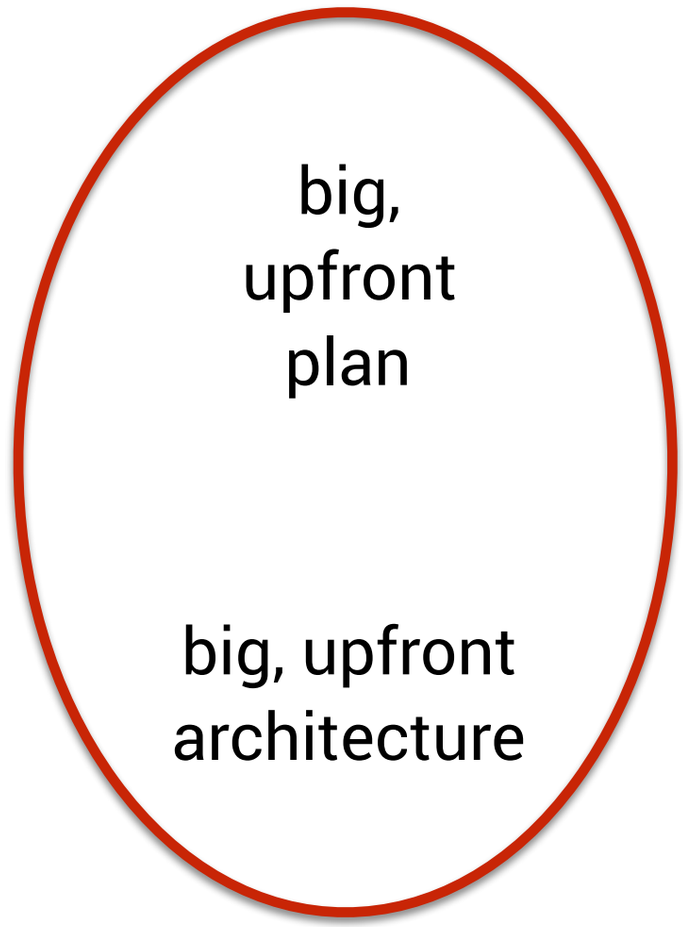


TDD

bad agile



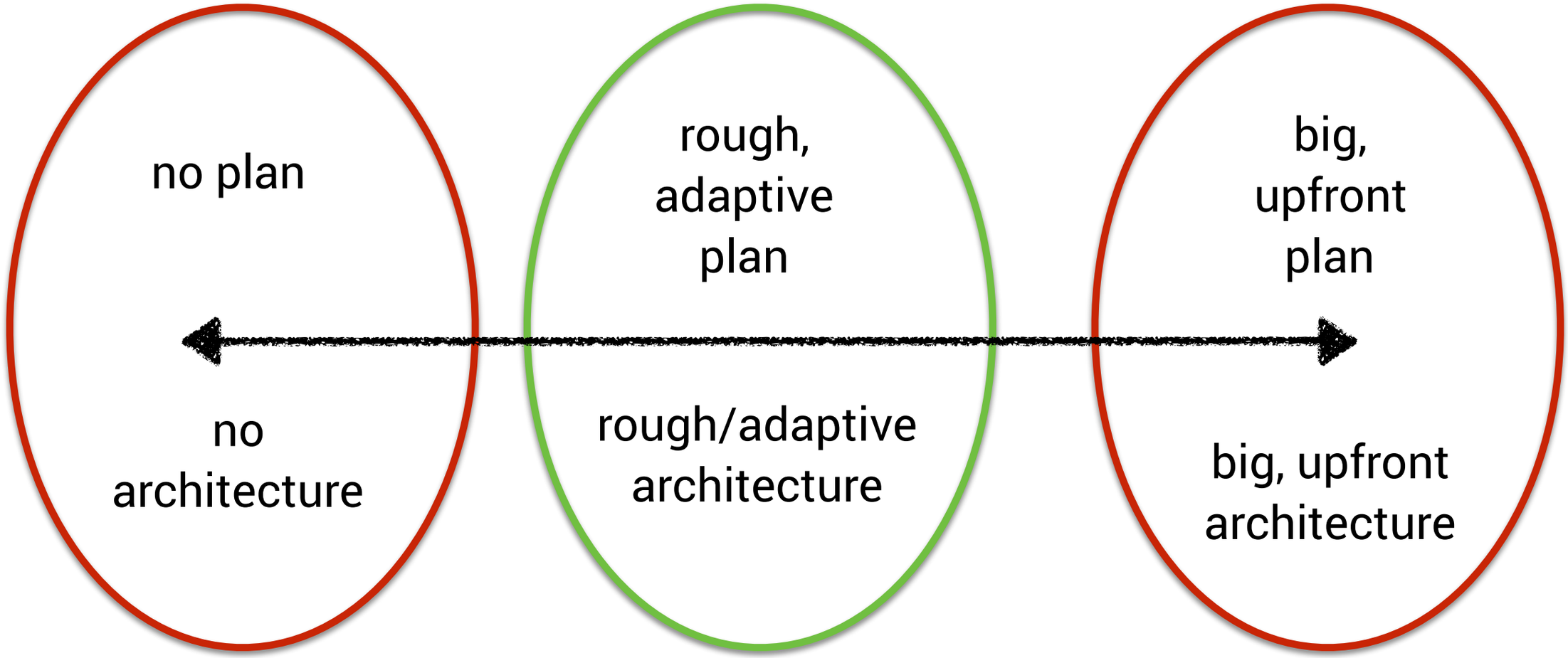
many traditional projects



bad agile

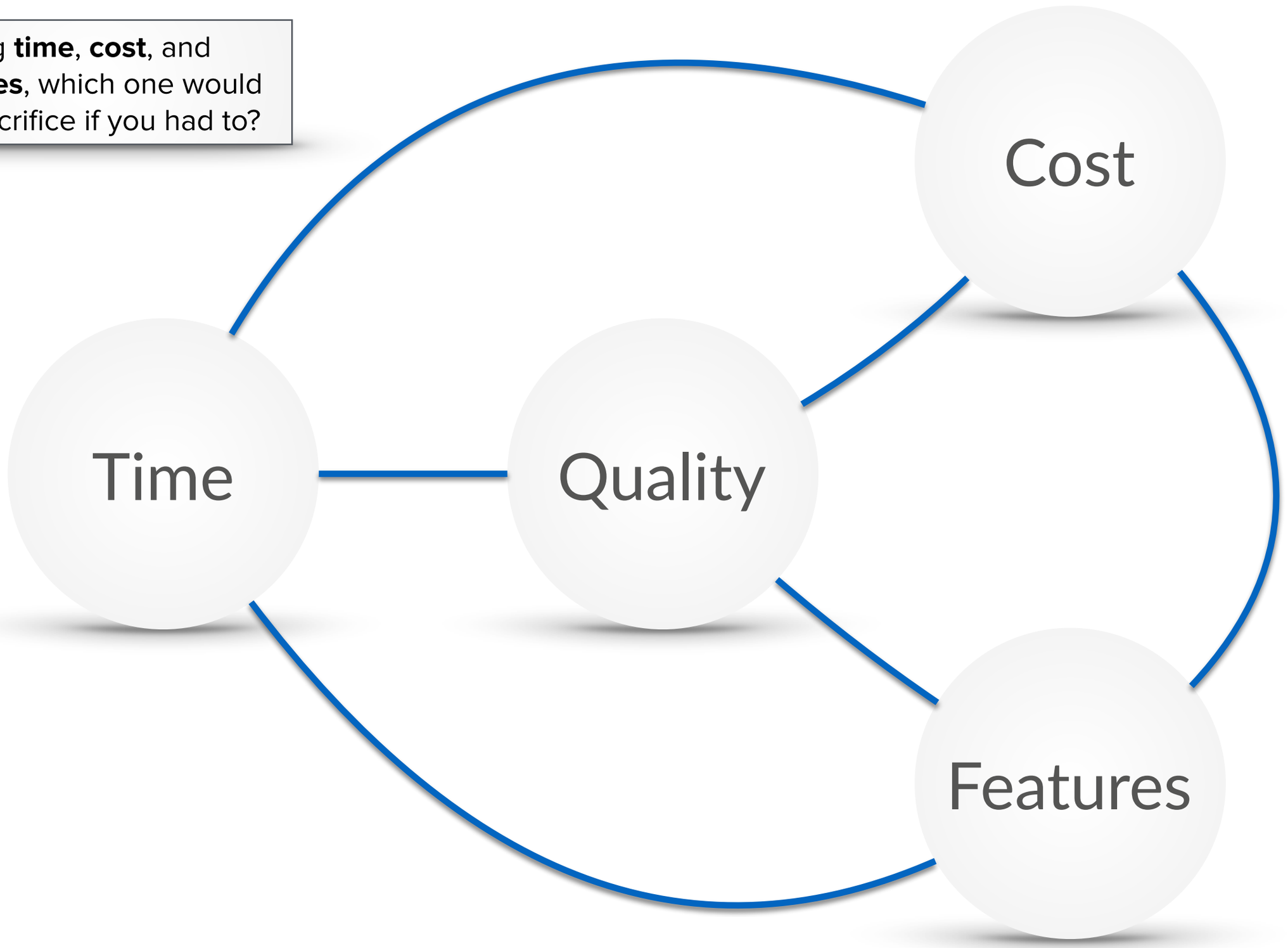
good agile

many traditional projects

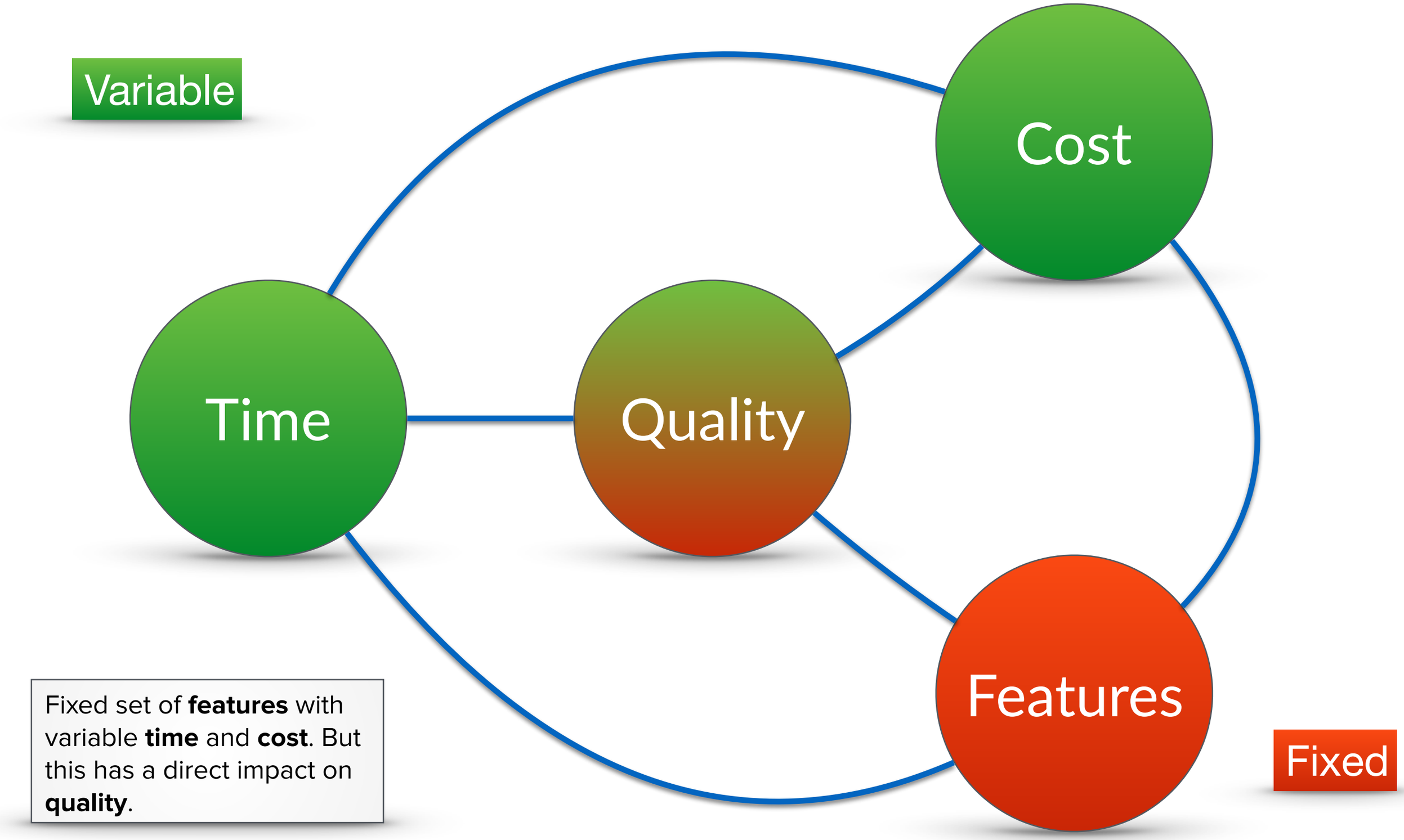


Quality

Among **time**, **cost**, and **features**, which one would you sacrifice if you had to?

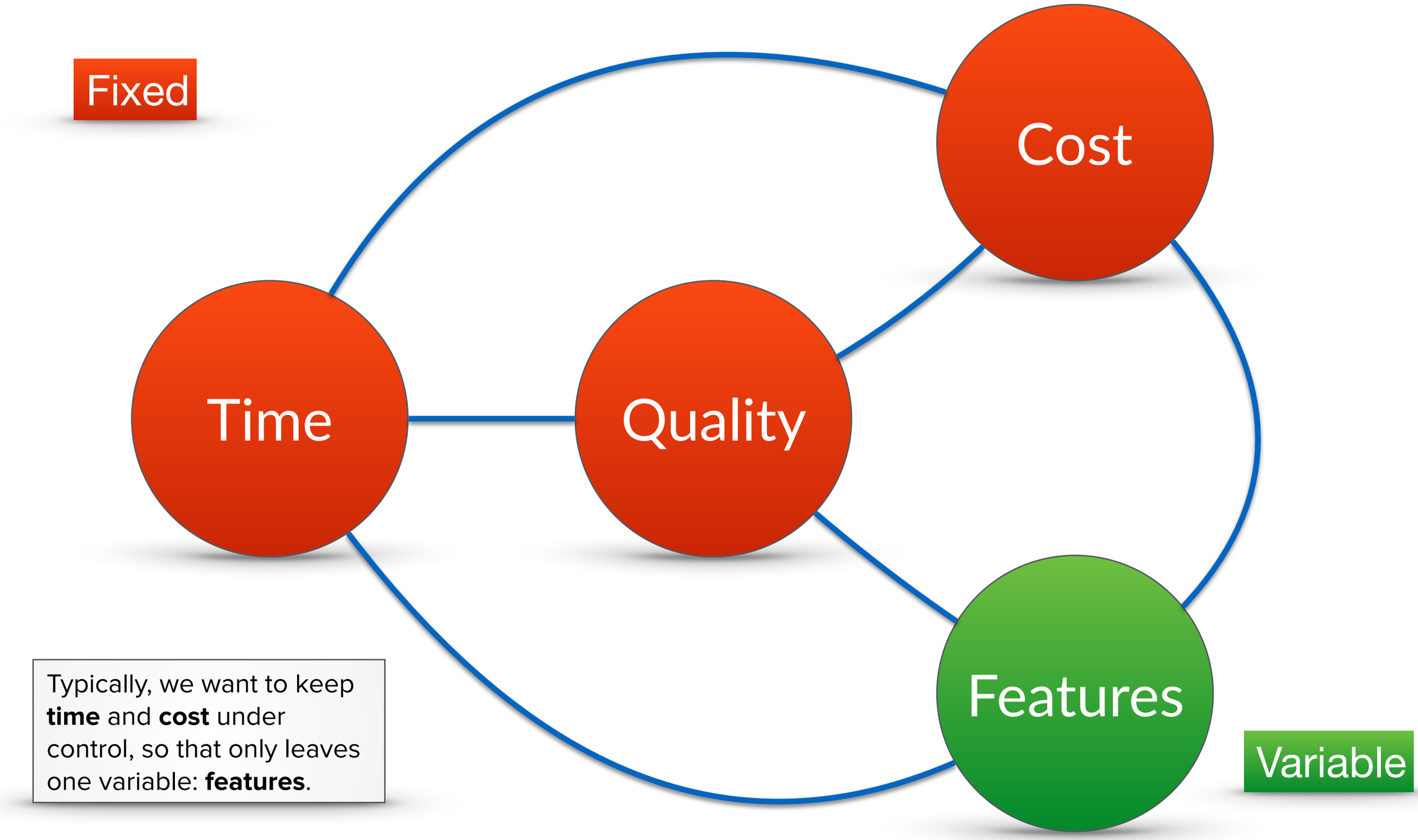


Quality - Traditional Approach



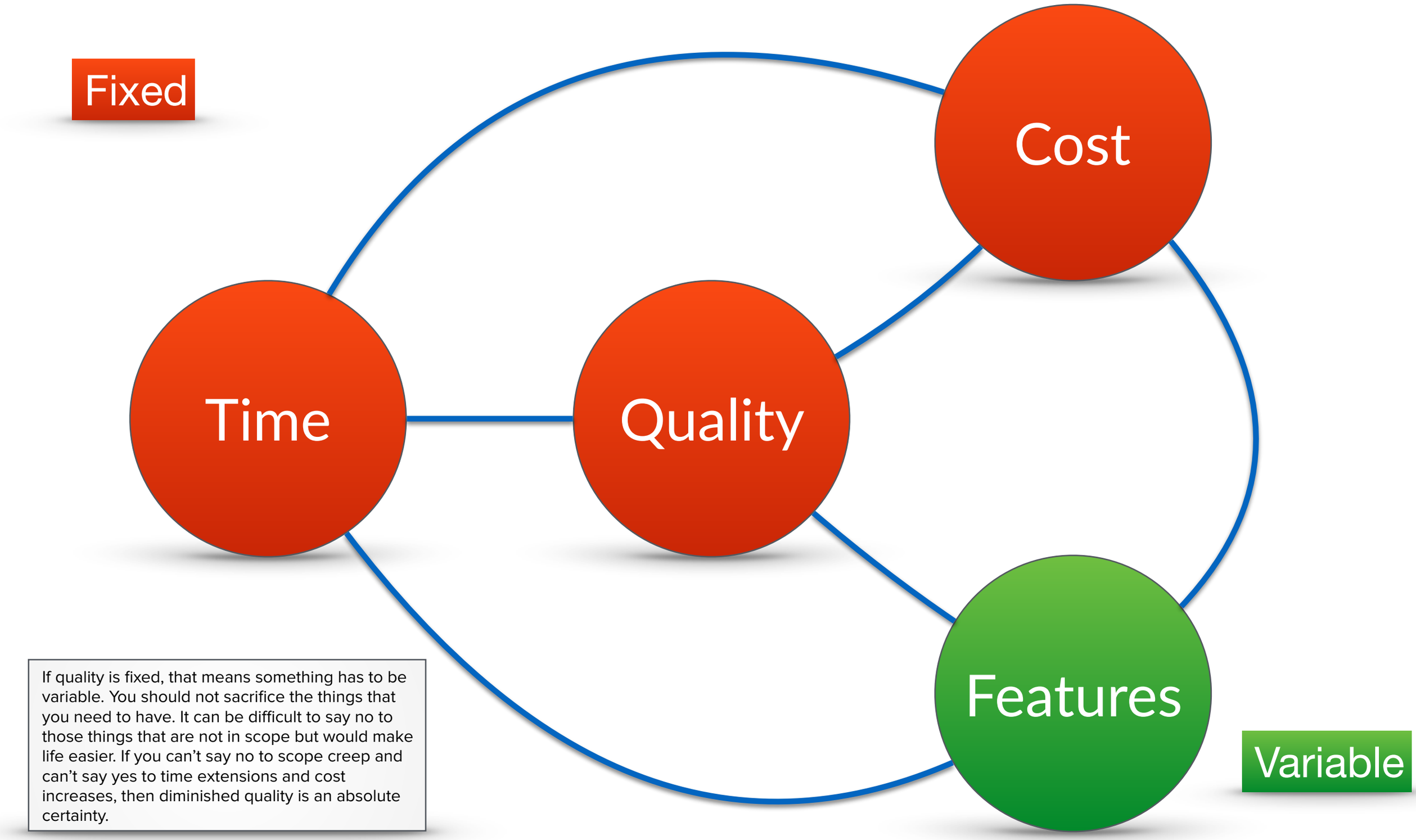
Fixed set of **features** with variable **time** and **cost**. But this has a direct impact on **quality**.

Quality - Agile Approach



Typically, we want to keep **time** and **cost** under control, so that only leaves one variable: **features**.

Quality - Agile Approach



If quality is fixed, that means something has to be variable. You should not sacrifice the things that you need to have. It can be difficult to say no to those things that are not in scope but would make life easier. If you can't say no to scope creep and can't say yes to time extensions and cost increases, then diminished quality is an absolute certainty.

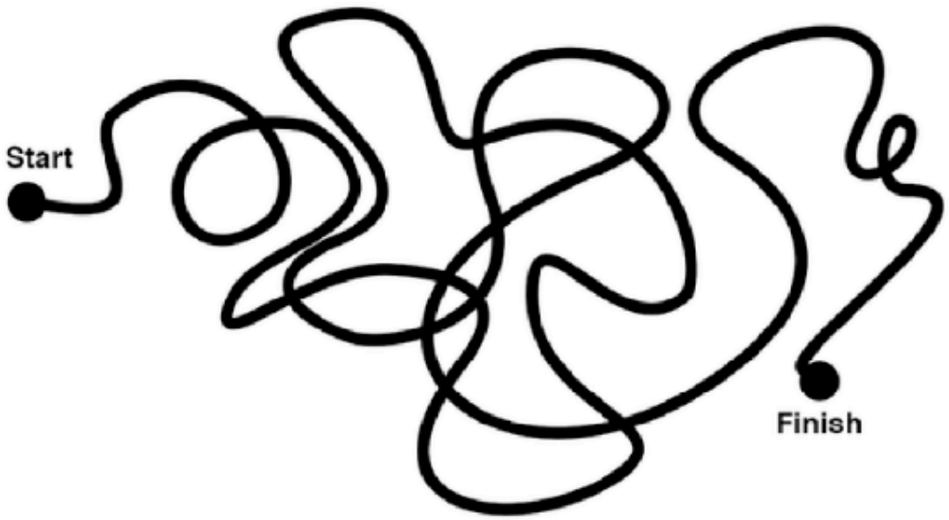
Advice

- Modularise: split into subsystems, layers or modules so that smaller chunks of functionality can be dealt with at a time.
- Iterate: “A complex system that works is invariably found to have evolved from a simple system that worked.” – John Gall
- Self-documenting code: The naming of the classes, methods and variables is incredibly important.
- No duplication
- Unit testing. It’s an easy way to ensure your smallest program parts work as expected, and you get repeatable tests that can be run again and again.
- Version control
- Write with other people in mind
- Plan for failure – logging and error handling.
- Track issues

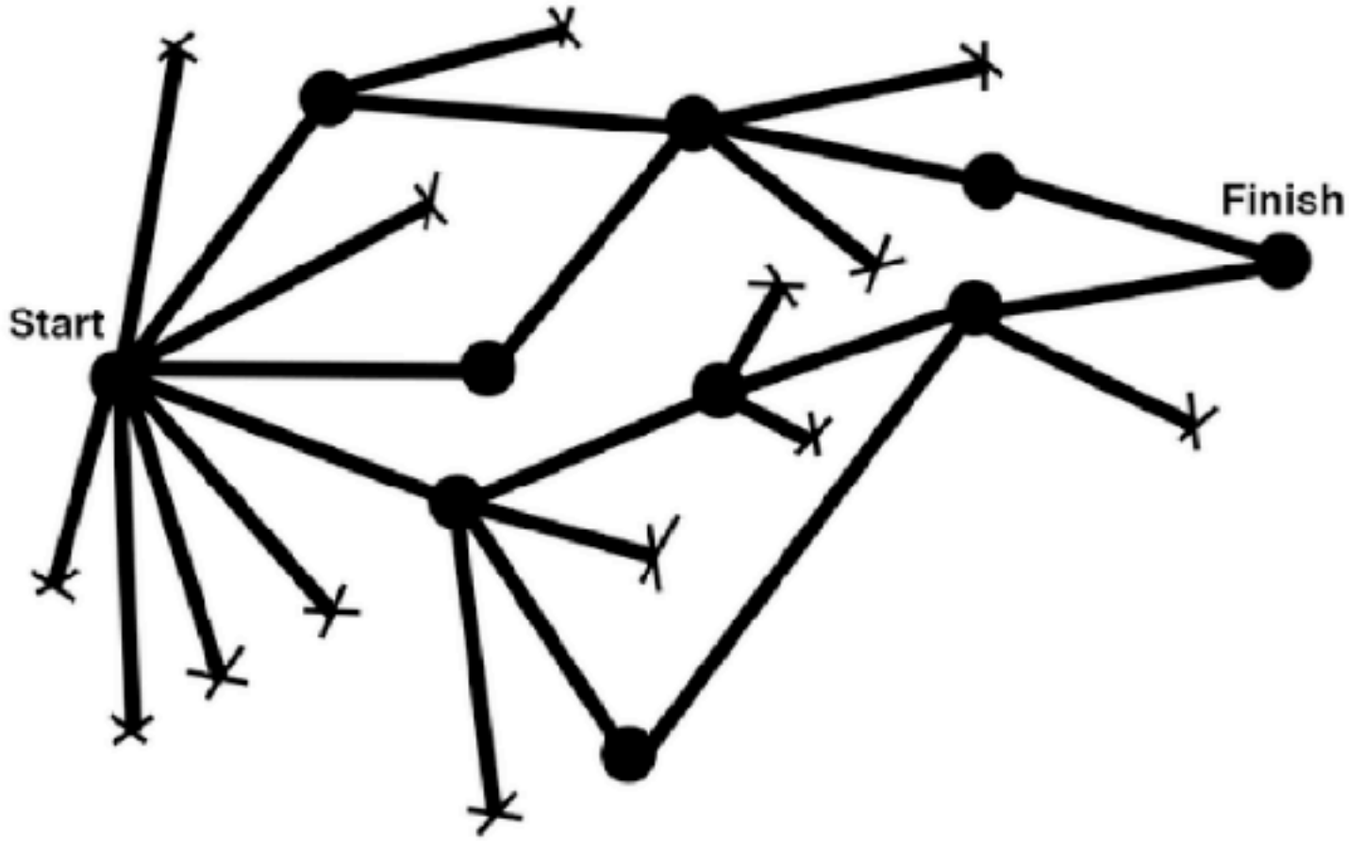
Advice - Clean Code

- Bad code does too much – Clean code is focused
- The language you wrote your code with should look like it was made for the problem
- It is not the language that makes a program look simple, but the programmer who makes the language appear simple.
- It should not be redundant
- Reading your code should be pleasant (DRY, YAGNI)
- Can be easily extended by any other developer
- It should have minimal dependencies
- Smaller is better
- It should have accompanying tests

Experience



How a Junior Solves Problem



How a Senior Solves Problem


ALICE, DO YOU HAVE ANY VALUABLE CAREER ADVICE?
WORK SO HARD THAT IT DESTROYS YOUR HEALTH AND CROWDS OUT ANY CHANCE OF HAVING A PERSONAL LIFE.
WOULDN'T THAT MAKE ME. . . UNHAPPY?
YOU DIDN'T ASK FOR HAPPINESS ADVICE.
Dilbert.com DilbertCartoonist@gmail.com
12-6-12 ©2012 Scott Adams, Inc.

That's all Folks!