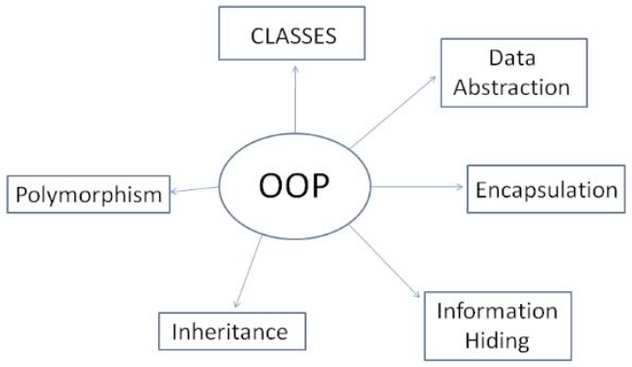
COMP30670
Software Engineering (Conversion)

13 - Project Planning
06/03/2017

Dr. Aonghus Lawlor

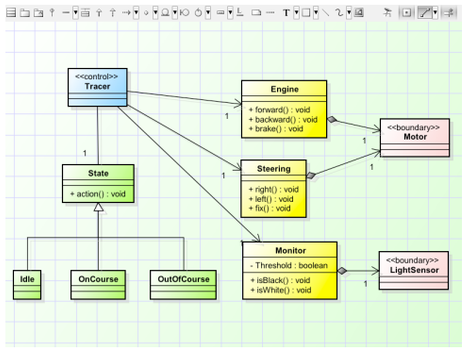
aonghus.lawlor@insight-centre.org

Future Areas to Cover

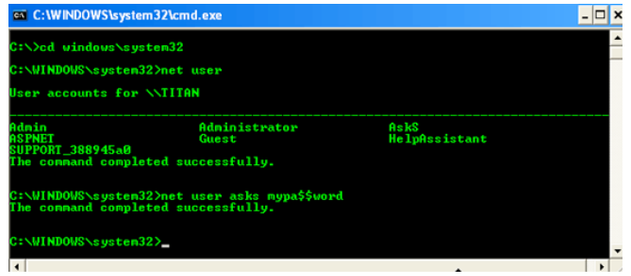


Object Oriented Python

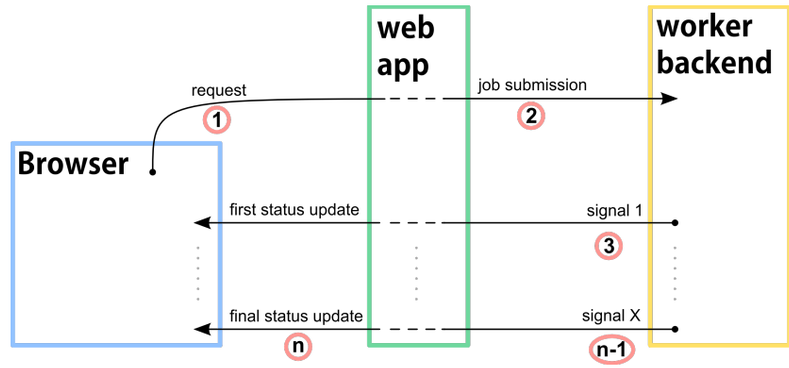
Python and Databases



Object Oriented Python



- **Basic Command Line Fu**



client/server backend

Agile Review

Agile Manifesto

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Scrum summary

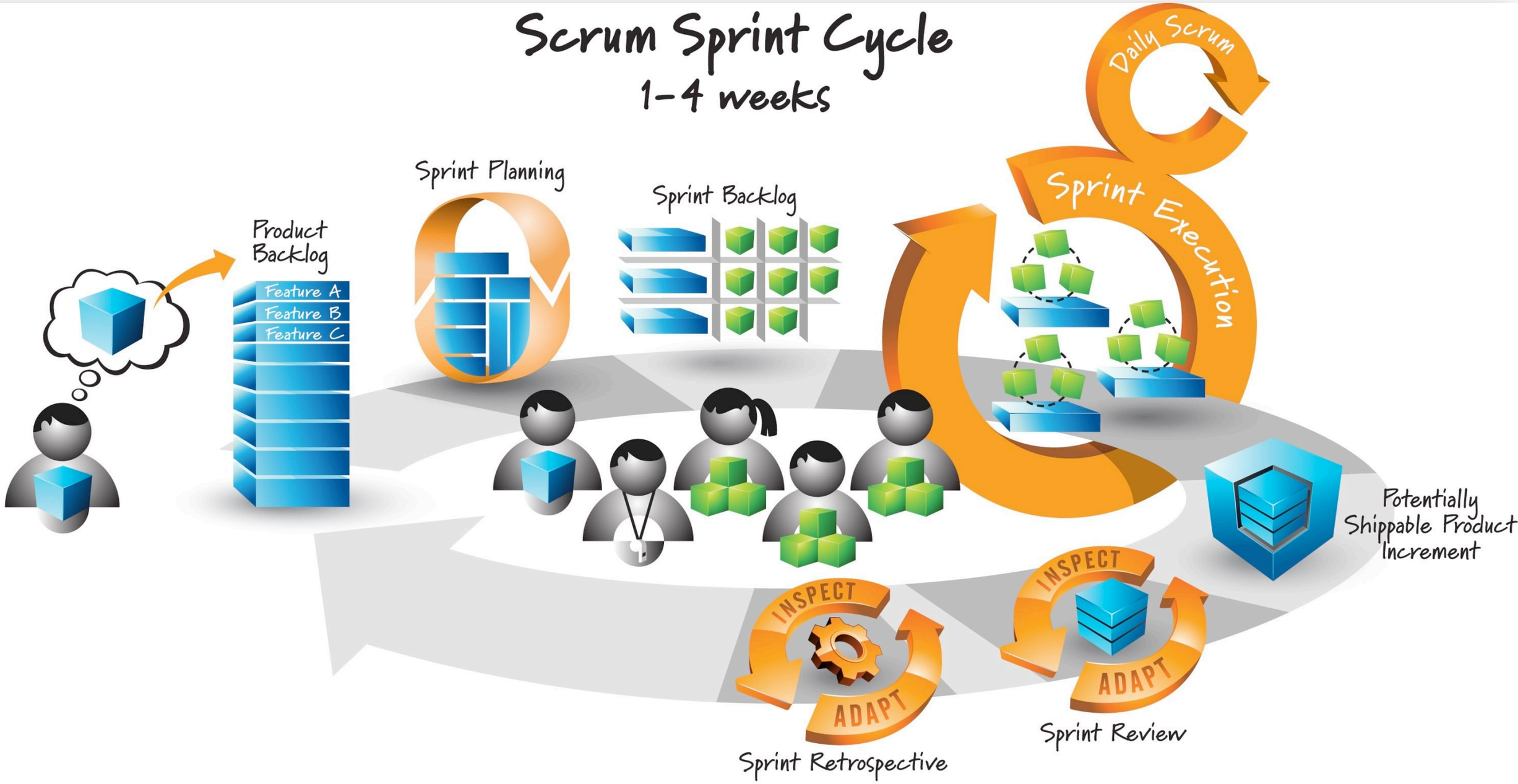
- kickoff,
- sprint planning meeting
- sprint
- daily Scrum (standup)
- sprint review meeting

Scrum summary

- **Sprint Planning:**
  - the group defines the product backlog- a list of the project requirements.
  - the group determines the sprint goal, which is the formal outcome(s) from this particular sprint
  - create the sprint backlog.

- **Kickoff meeting**
  - the group define the high-level backlog for the project and the major project goals.

Scrum


Scrum Sprint Cycle
1-4 weeks
Product Backlog
Feature A
Feature B
Feature C
Sprint Planning
Sprint Backlog
Daily Scrum
Sprint Execution
Potentially Shippable Product Increment
INSPECT
ADAPT
Sprint Retrospective
INSPECT
ADAPT
Sprint Review

Scrum Summary

Each team member answers 3 questions:

- What did you do since the last standup?
- What are you doing until the next standup?
- What is stopping you getting on with your work?

not a problem solving session and is not really designed to be a way of collecting information about who (or what) is behind schedule

track progress of team and allow team members to make commitments to each other and the Scrum master so that work can proceed in the best way.

Scrum Summary

- **Sprint review meeting is held at the end of each sprint.**
- **During the meeting, the functionality that was created during the sprint is demonstrated to the product owner (demonstrator)**

Scrum Summary

product backlog

requirements for the project expressed as a prioritised list of backlog items

cannot be changed until the next sprint planning meeting

sprint backlog

subset of product backlog items that are defined as part of the work for a particular sprint

usually internal to the team (does not include the product owner)

burn down charts

provide information in an easy to comprehend manner

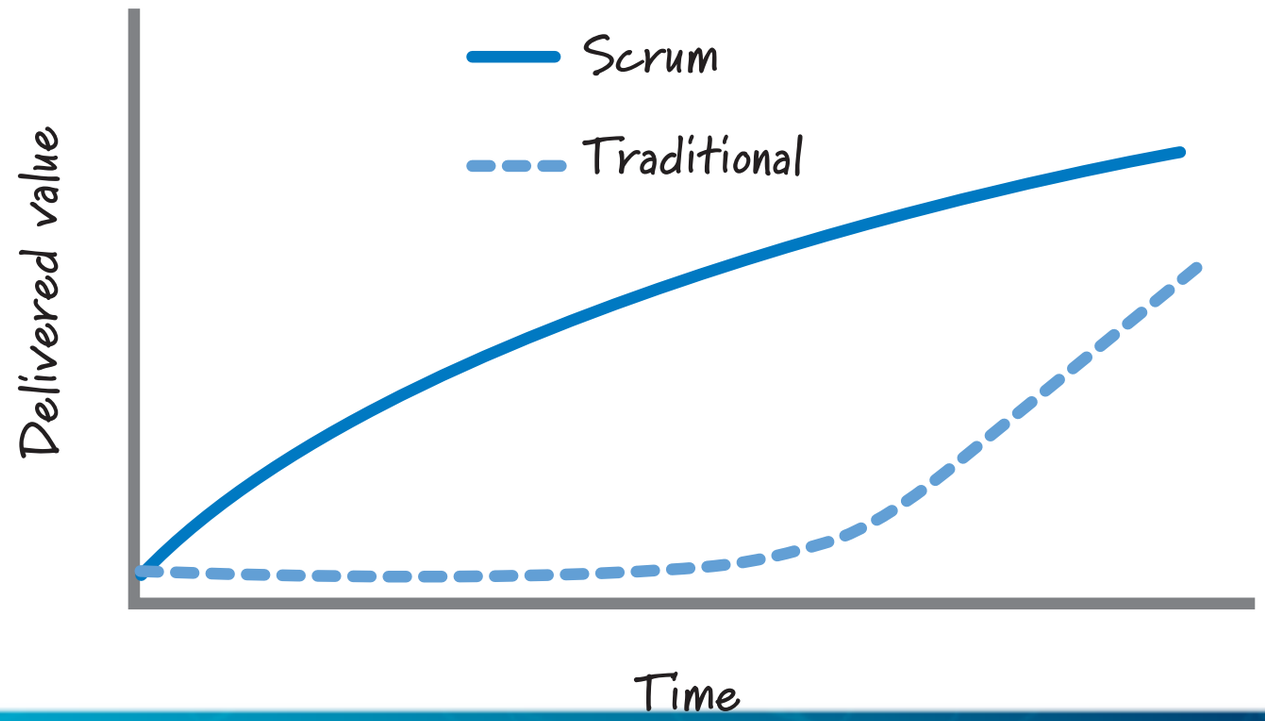
each task is typically represented in terms of time (x-axis) and duration (y-axis)

Scrum Summary

- each team members bears major responsibility for their part of the project, ownership of the project is more broadly based.
- requires extensive communication, which helps teams organise more effectively.
- ultimately, can lead to greater productivity for everyone involved.
- frequent communication does not mean that no documentation is required
- without adequate oversight, a project could degenerate into “undisciplined hacking” because documentation is not maintained.

Scrum

- Plan-driven, sequential development focuses on diligently following the process
- Scrum is a customer-value-centric form of development
- based on a prioritised, incremental model of delivery
- highest-value features are continuously built and delivered in the next iteration
- As a result, customers get a continuous flow of high-value features sooner.



Performance

- **Go fast- but never hurry.**
- **Build in quality.**
- **Employ minimally sufficient ceremony.**

Performance

- Plan-driven development: if we follow the plan and do things right the first time, we'll avoid costly and time-consuming rework.
- Moving from step to step quickly is desirable, but not the principal goal.
- core goals: nimble, adaptable, and speedy.
- time is of the essence, but don't rush to get things done.
- sustainable pace—people should be able to work at a pace that they can continue for an extended period of time (hurrying will likely come at the expense of quality)

Quality

- plan-driven development: through careful, sequential performance of work we get a high-quality product- we can't actually verify this quality until we do the testing of the integrated product, which occurs during a late phase of the process.

- Scrum: quality isn't something a testing team "tests in" at the end

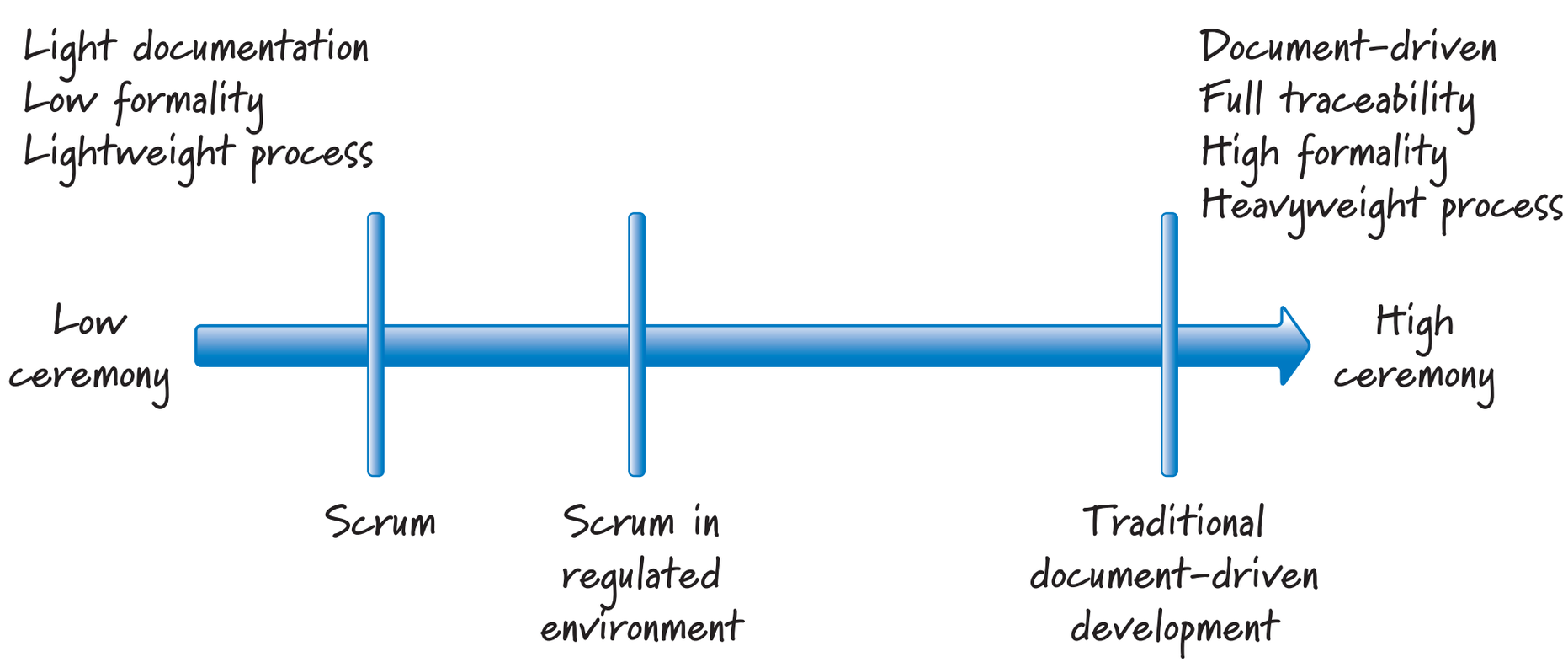
- it is something that a cross-functional Scrum team owns and continuously builds in and verifies every sprint

- each increment of value that is created is completed to a high level of confidence

- the need for any significant late testing to tack on quality is substantially reduced.

Ceremony

- one that is minimally sufficient is good enough
- what constitutes minimally sufficient can differ
- we are trying to avoid is work that adds no short-term or long-term value



Requirements

- **sequential product development**
  - requirements are treated much as they are in manufacturing
  - they are required, nonnegotiable specifications to which the product must conform
  - the requirements are created up front and given to the development group in the form of a highly detailed document (SRS)
  - It is the job of the development group, then, to produce a product that conforms to the detailed requirements.

Requirements

Customer: “Now that I see these built features, I realise I need this other feature that isn’t in the requirements document.”

Developers: “If you wanted that feature, why didn’t you specify it up front?”

Customer: “Well, I didn’t realise I needed that feature until I saw the product coming together.”

Developers: “Well, if you had thought longer and harder about the requirements up front, you would have found that feature then instead of now.”

Requirements

- **Agile/Scrum**
  - requirements are an important degree of freedom that we can manipulate to meet our goals
  - for example, if we're running out of time or money, we can drop low-value requirements
  - If new information indicates that the cost/benefit ratio of a requirement has become significantly less favourable, we can choose to drop it
  - if a new high-value requirement emerges, we have the ability to add it to the product, perhaps discarding a lower-value requirement to make room.

User Stories

- **How to represent product backlog items:**
  - User stories are a convenient format for expressing the desired value for many types of product backlog items, especially features.
  - User stories are crafted in a way that makes them understandable to both business people and technical people.
  - They are structurally simple and provide a great placeholder for a conversation.
  - Additionally, they can be written at various levels of granularity and are easy to progressively refine.

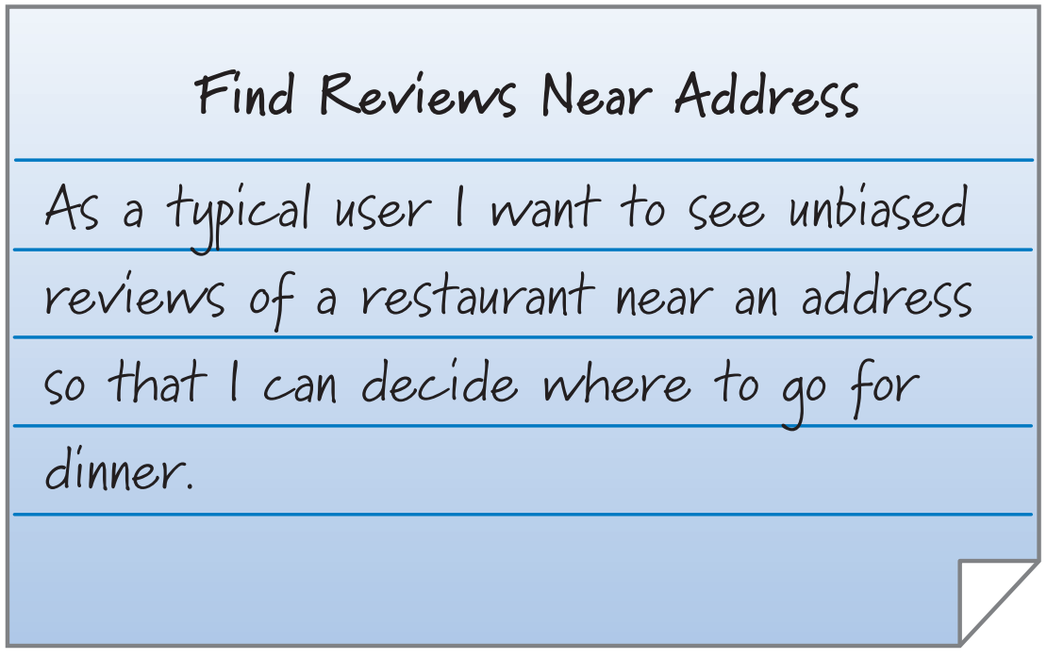
User Stories

"As a _____, I want ____, so that _____

User stories are part of an agile approach that helps shift the focus from writing about requirements to talking about them. All agile user stories include a written sentence or two and, more importantly, a series of conversations about the desired functionality.

User Stories

User stories are part of an agile approach that helps shift the focus from writing about requirements to talking about them. All agile user stories include a written sentence or two and, more importantly, a series of conversations about the desired functionality.



User Stories

- A user story also contains confirmation information in the form of conditions of satisfaction.
- These are acceptance criteria that clarify the desired behaviour.
- They are used by the development team to better understand what to build and test and by the product owner to confirm that the user story has been implemented to his satisfaction.

User Stories

Upload File

As a wiki user I want to upload a file to the wiki so that I can share it with my colleagues.

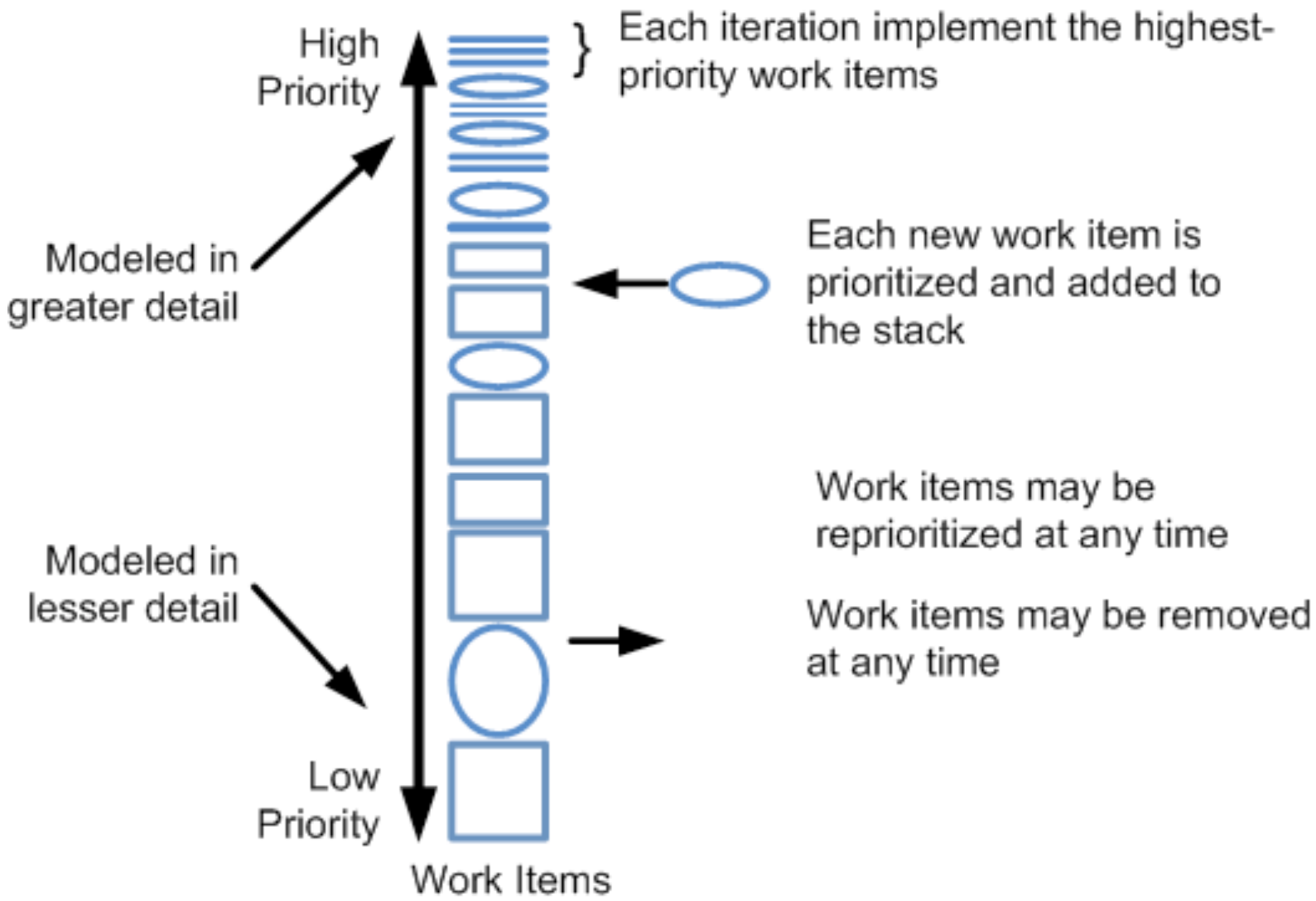
Conditions of Satisfaction

Verify with .txt and .doc files
Verify with .jpg, .gif, and .png files
Verify with .mp4 files <= 1 GB
Verify no DRM-restricted files

User Stories

- User stories are simple enough that people can learn to write them in a few minutes
- Use the simplest tool. User stories are often written on index cards
- Remember non-functional requirements. Stories can be used to describe a wide variety of requirements types. (eg. “The Students can purchase parking passes online” this user story is a usage requirement similar to a use case whereas “The Transcripts will be available online via a standard browser” is closer to a technical requirement.
- Indicate the estimated effort.
- Indicate the priority. I often use a scale of one to ten with one being the highest priority

User Stories



Technical Debt

"Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite. . . . The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organisations can be brought to a stand-still under the debt load of an unconsolidated implementation. . . ."

Ward Cunningham 1992

Technical Debt

- **Nowadays, technical debt refers both to the shortcuts we purposely take and also to the many bad things that plague software systems:**
  - Unfit/bad design—a design that once made sense but no longer does, given important changes to the business or technologies we now use
  - Defects—known problems in the software that we haven't yet invested time in removing
  - Insufficient test coverage- areas where we know we should do more testing but don't
  - Excessive manual testing—testing by hand when we really should have automated tests

Technical Debt

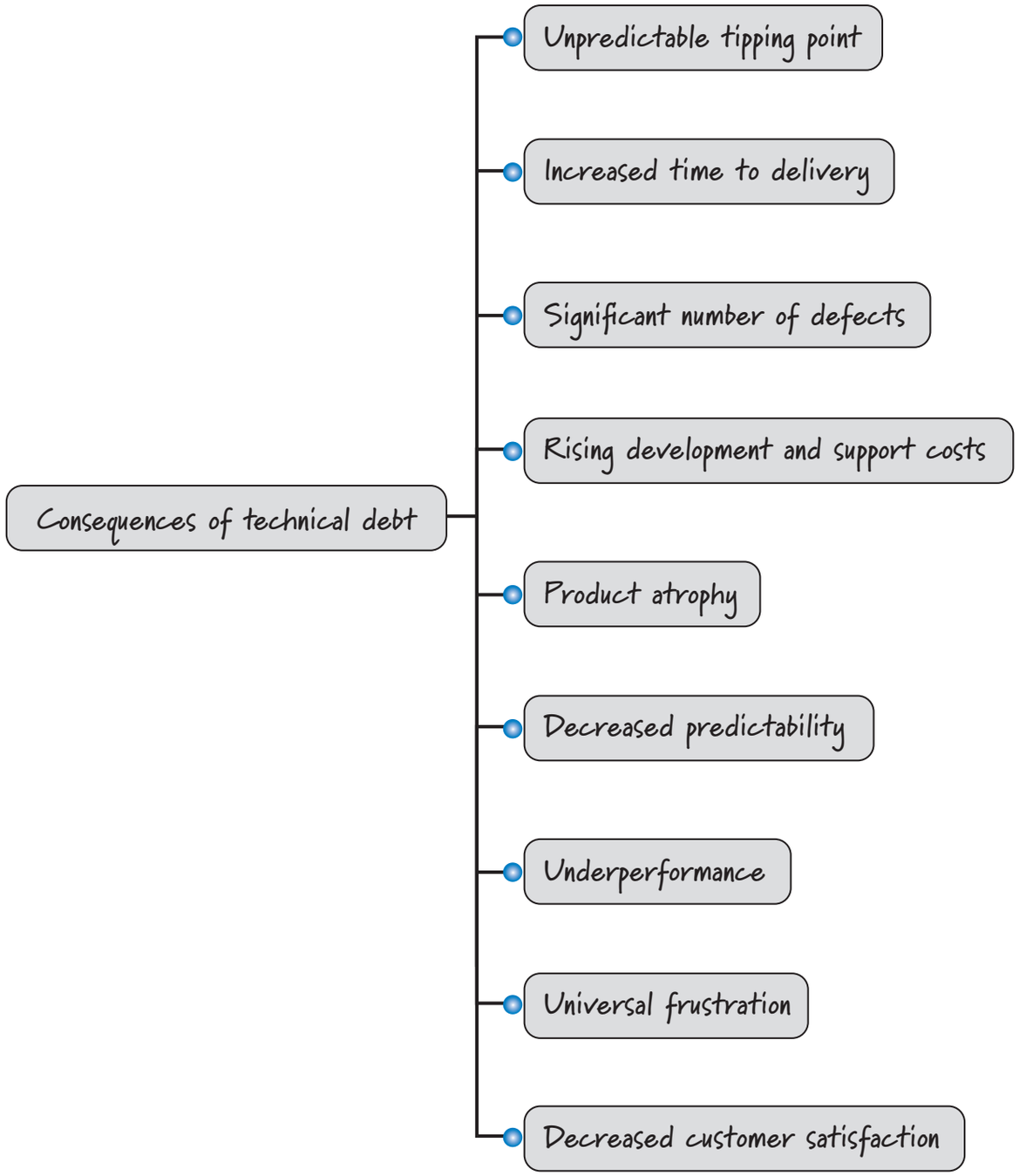
- our understanding of what makes for a good design emerges from doing the design work and building user-valuable features on it
- we can’t perfectly predict up front how our product and its design will need to evolve over time
- design and implementation decisions we made early on might need to change as we close important learning loops and acquire validated learning
- he changes required in the affected areas are *unavoidable technical debt*.

Technical Debt

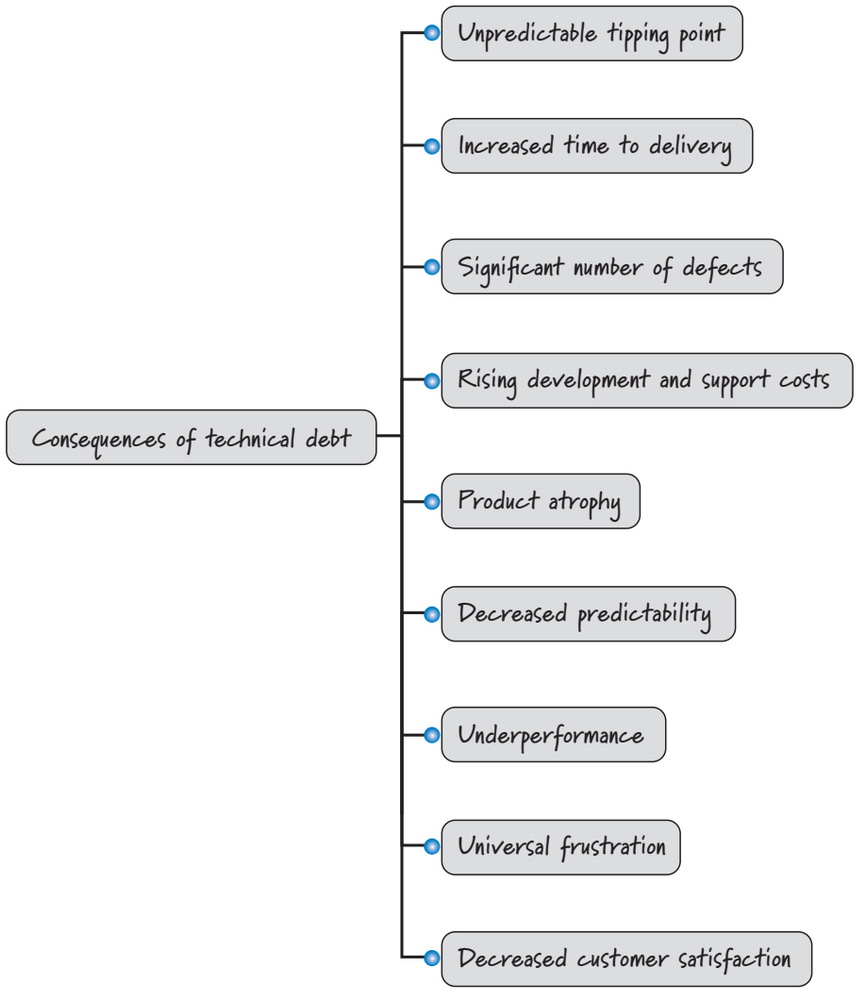
- say we licensed a third-party component for use in our product and the interfaces to that component evolve over time.
- our product that once functioned well with the third-party component accrues technical debt through no fault of our own
- although this debt might be predictable (it's not unreasonable to assume that the vendor will change its component interfaces over time), it's not preventable because we can't foresee how the component developers might evolve the component in the future

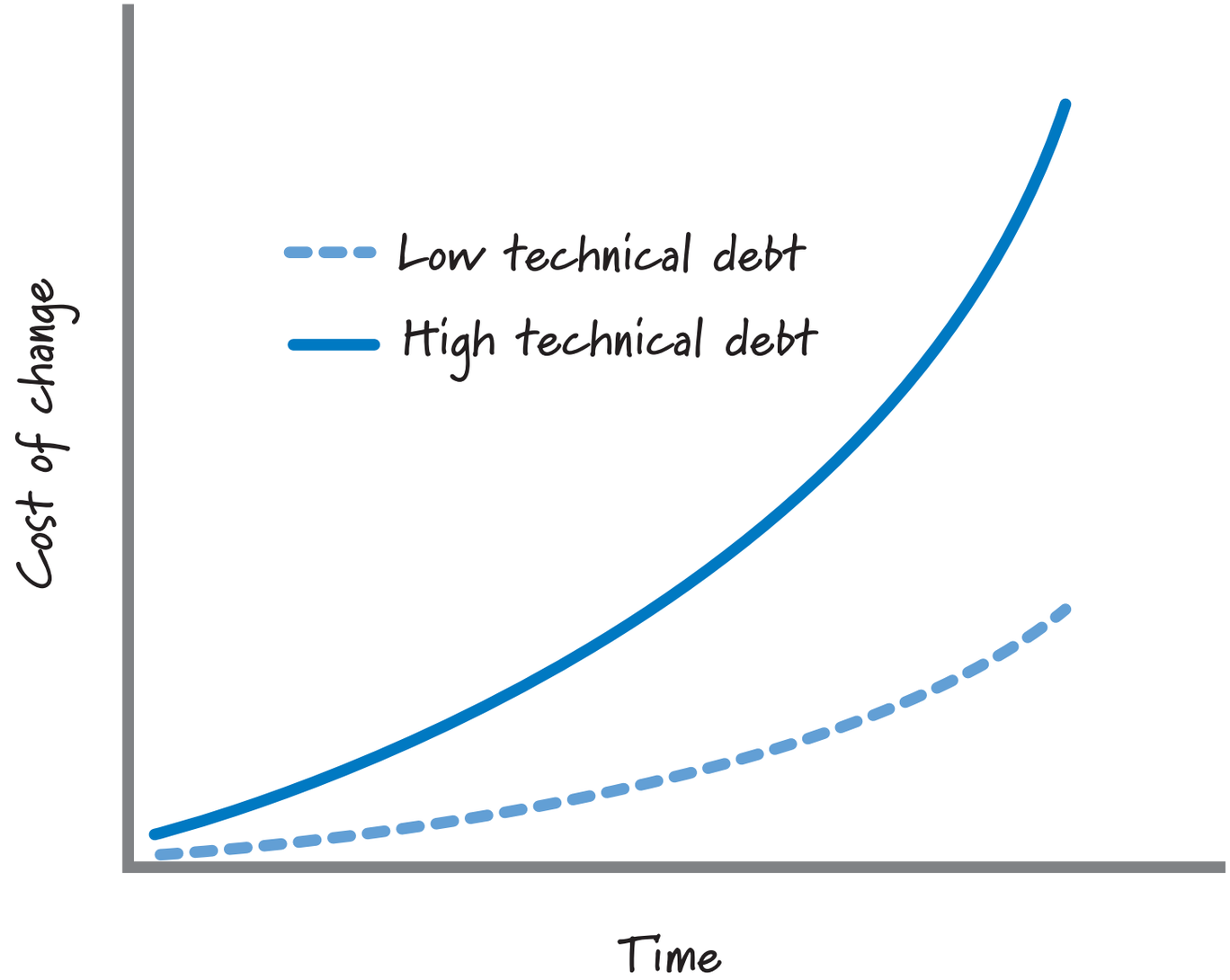
Technical Debt

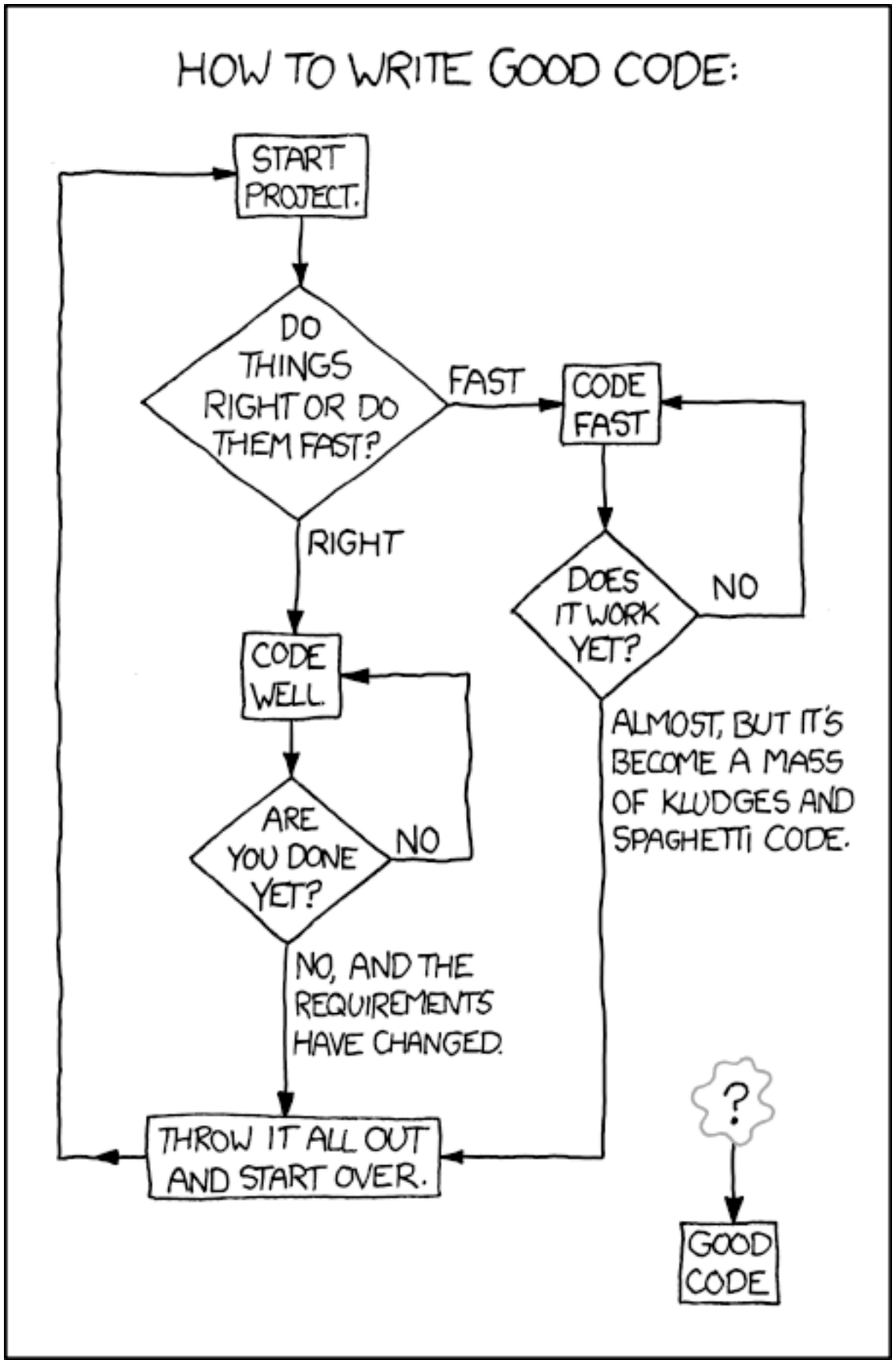
- just like financial debt, technical debt requires interest payments, which come in the form of extra future development effort
- we can choose to continue paying the interest (by working around the problems), or we can pay down the debt principal (for example, by refactoring the code to make it cleaner and easier to modify)



Technical Debt



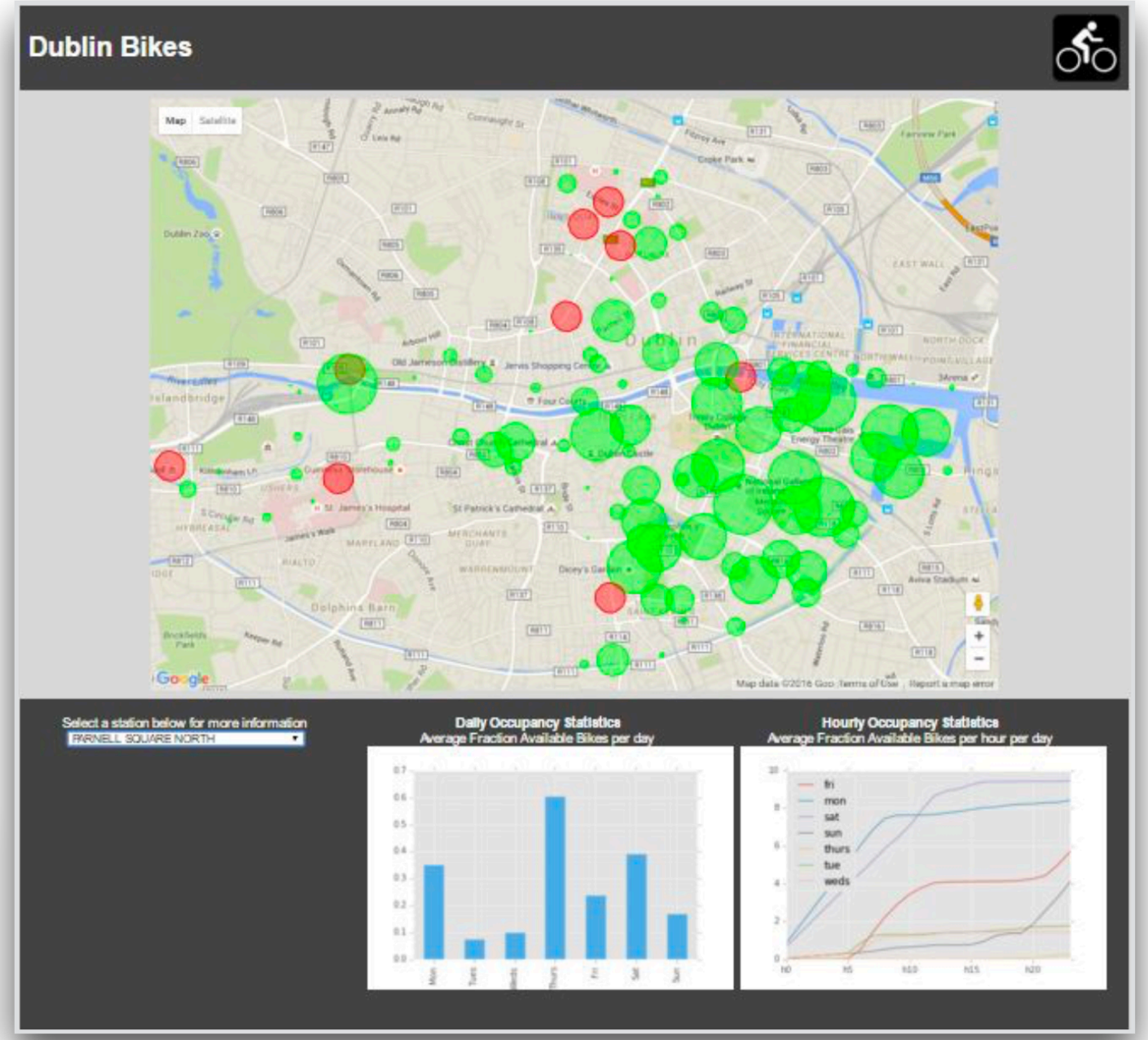



HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
DOES IT WORK YET?
NO
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
RIGHT
CODE WELL.
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
THROW IT ALL OUT AND START OVER.
?
GOOD CODE

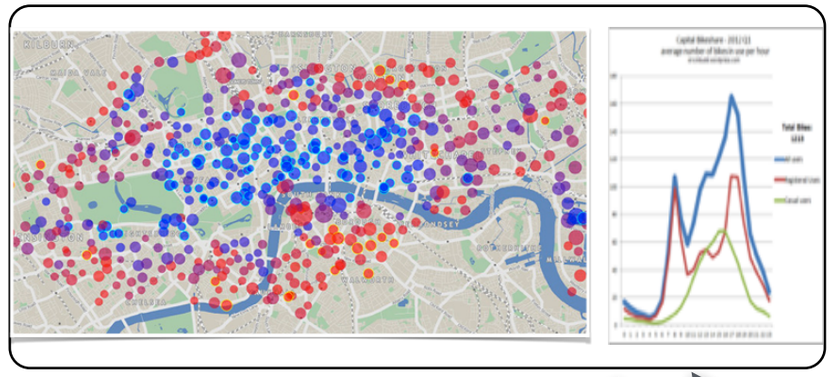
Dublin Bikes

COMP30670 Project

work samples



client



requests

JCDecaux developer

python

```
form="""
<form method="post" action="/testform">
    <input name="q">
    <input type="submit">
</form>
"""

class MainPage(webapp2.RequestHandler):
    def get(self):
        #self.response.headers['Content-Type'] = 'text/plain'
        self.response.out.write(form)

class TestHandler(webapp2.RequestHandler):
    def post(self):
        q = self.request.get("q")
        self.response.out.write(q)

        #self.response.headers['Content-Type'] = 'text/plain'
        #self.response.out.write(self.request)
```

Python code to scrape station and occupancy data from online source

Data Store

store data in a database (SQLite or MongoDB)

python

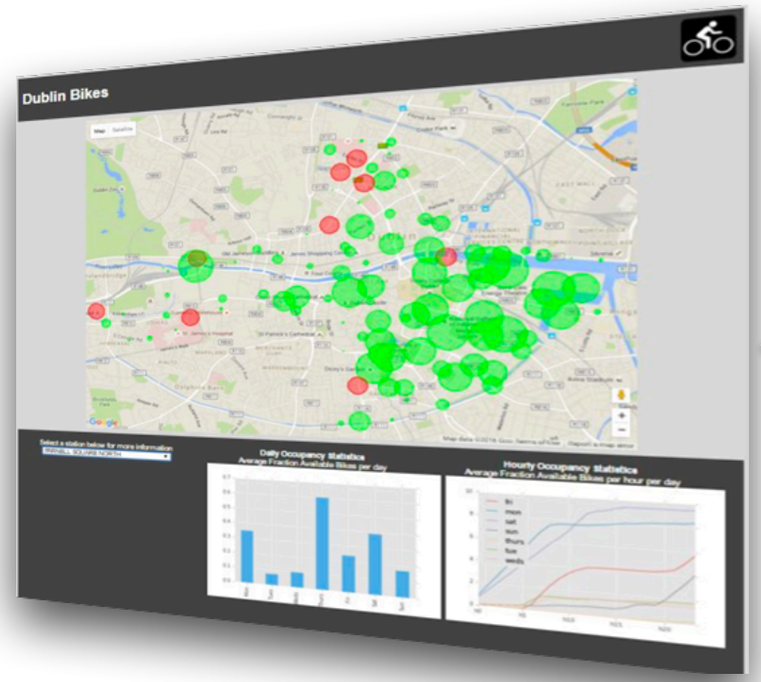
```
form="""
<form method="post" action="/testform">
    <input name="q">
    <input type="submit">
</form>
"""

class MainPage(webapp2.RequestHandler):
    def get(self):
        #self.response.headers['Content-Type'] = 'text/plain'
        self.response.out.write(form)

class TestHandler(webapp2.RequestHandler):
    def post(self):
        q = self.request.get("q")
        self.response.out.write(q)

        #self.response.headers['Content-Type'] = 'text/plain'
        #self.response.out.write(self.request)
```

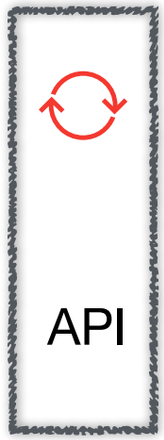
Python code to analyse data and provide occupancy information

server

AWS

COMP30670 Project

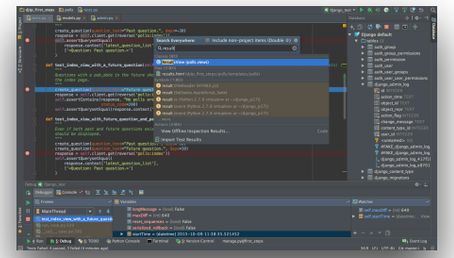




API's are implemented with python *flask* app

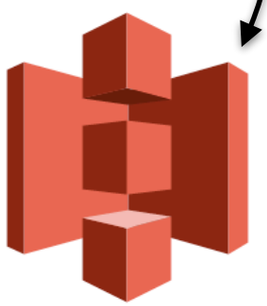
push to github

pull from github



develop on your local machine

Amazon EC2



Amazon S3

store in S3

Database Services

History

EC2

RDS

Console Home

S3

VPC

Billing

Search services

Group

Compute
EC2
EC2 Container Service
Lightsail
Elastic Beanstalk
Lambda
Batch

Storage
S3
EFS
Glacier
Storage Gateway

Database
RDS
DynamoDB
ElastiCache
Redshift

Networking & Content Delivery
VPC
CloudFront
Direct Connect
Route 53

Migration
Application Discovery Service
DMS
Server Migration
Snowball

Developer Tools
CodeCommit
CodeBuild
CodeDeploy
CodePipeline

Management Tools
CloudWatch
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Trusted Advisor
Managed Services

Security, Identity & Compliance
IAM
Inspector
Certificate Manager
Directory Service
WAF & Shield
Compliance Reports

Analytics
Athena
EMR
CloudSearch
Elasticsearch Service
Kinesis
Data Pipeline
QuickSight

Artificial Intelligence
Lex
Polly
Rekognition
Machine Learning

Internet Of Things
AWS IoT

Game Development
GameLift

Mobile Services
Mobile Hub
Cognito
Device Farm
Mobile Analytics
Pinpoint

Application Services
Step Functions
SWF
API Gateway
Elastic Transcoder

Messaging
SQS
SNS
SES

Business Productivity
WorkDocs
WorkMail
Amazon Chime

Desktop & App Streaming
WorkSpaces
AppStream 2.0

Step 1: Select Engine

Step 2: Specify DB Details

Step 3: Configure Advanced Settings

Your current selection is eligible for the free tier.

Learn More.

Estimate your monthly costs for the DB Instance using the RDS Instance Cost Calculator.

Specify DB Details

Free Tier

The Amazon RDS Free Tier provides a single db.t2.micro instance as well as up to 20 GB of storage, allowing new AWS customers to gain hands-on experience with Amazon RDS. Learn more about the RDS Free Tier and the instance restrictions here.

Only show options that are eligible for RDS Free Tier

Instance Specifications

DB Engine	mariadb
License Model	general-public-license
DB Engine Version	MariaDB 10.0.24
DB Instance Class	db.t2.micro — 1 vCPU, 1 GiB RAM
Multi-AZ Deployment	No
Storage Type	Magnetic
Allocated Storage*	5 GB

Settings

DB Instance Identifier*	BikesDB
Master Username*	aonghus
Master Password*	•••••••••
Confirm Password*	•••••••••

Retype the value you specif for Master Password.

* Required

Cancel Previous Next Step

Step 2: Specify DB Details

Step 3: Configure Advanced Settings

Network & Security

VPC* Default VPC (vpc-a6b9dac3)

Subnet Group default

Publicly Accessible No

Availability Zone eu-west-1a

VPC Security Group(s)
Create new Security Group
default (VPC)
launch-wizard-1 (VPC)
launch-wizard-2 (VPC)

Database Options

Database Name dublinbikes

Database Port 3306

DB Parameter Group default.mariadb10.0

Option Group default:mariadb-10-0

Copy Tags To Snapshots

Enable Encryption No

Backup

Backup Retention Period 7 days

Backup Window No Preference

The daily time range (in UTC) during which automated backups are created if automated backups are enabled.

Monitoring

Enable Enhanced Monitoring No

Maintenance

Auto Minor Version Upgrade Yes

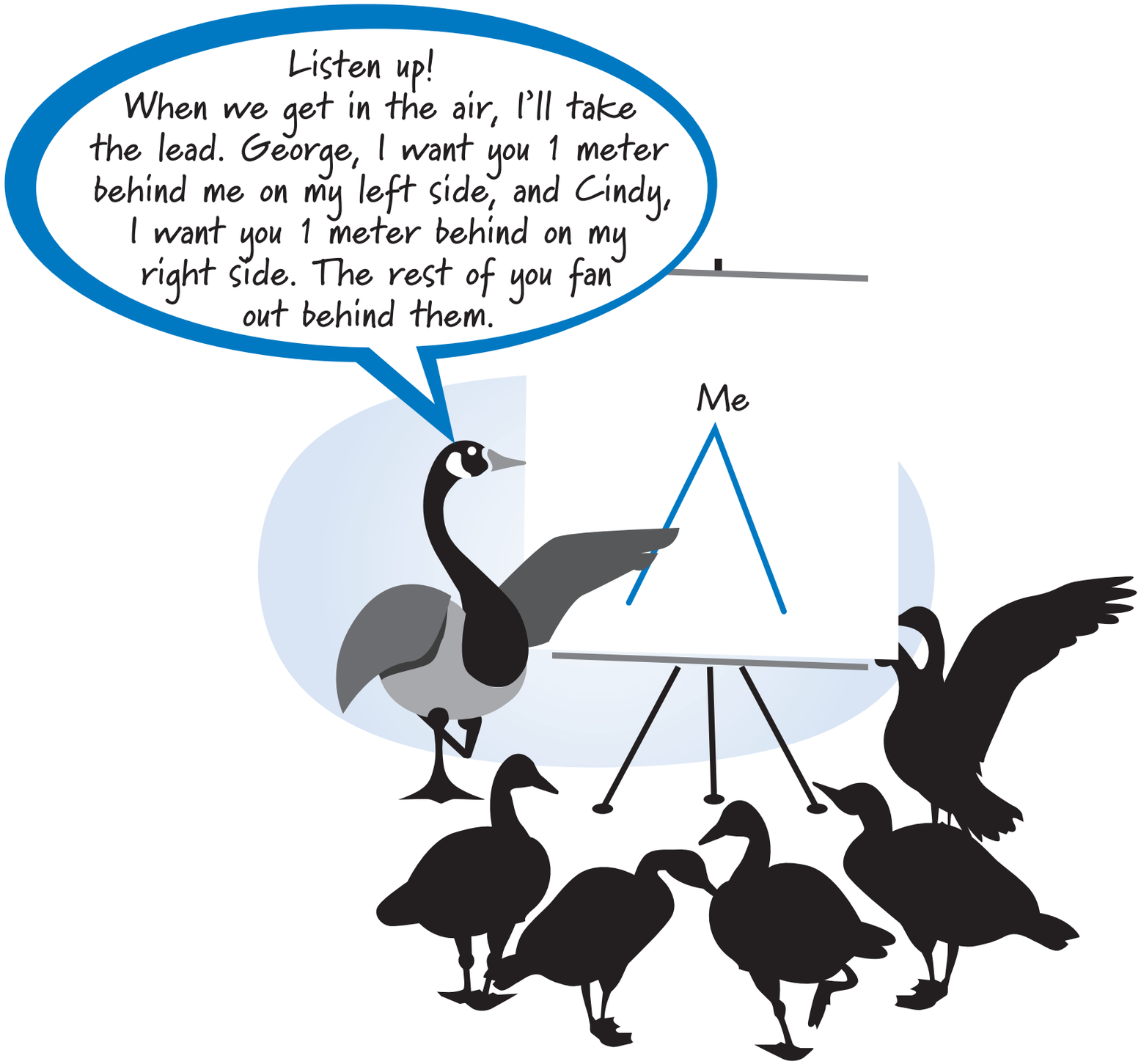
Maintenance Window No Preference

* Required

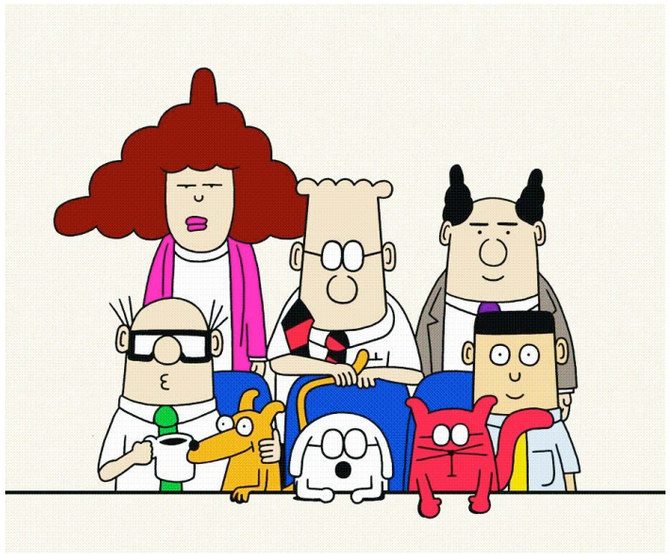
Cancel | Previous | Launch DB Instance

```
#!/usr/bin/python
import sys
import logging
import rds_config
import pymysql

rds_host  = rds_config.db_endpoint
name = rds_config.db_username
password = rds_config.db_password
db_name = rds_config.db_name
port = 3306

logger = logging.getLogger()
logger.setLevel(logging.INFO)

try:
    conn = pymysql.connect(rds_host, user=name,
                           passwd=password, db=db_name, connect_timeout=5)
except:
    logger.error("ERROR: Unexpected error: Could not connect to MySql instance.")
    sys.exit()
```

Top-down planning

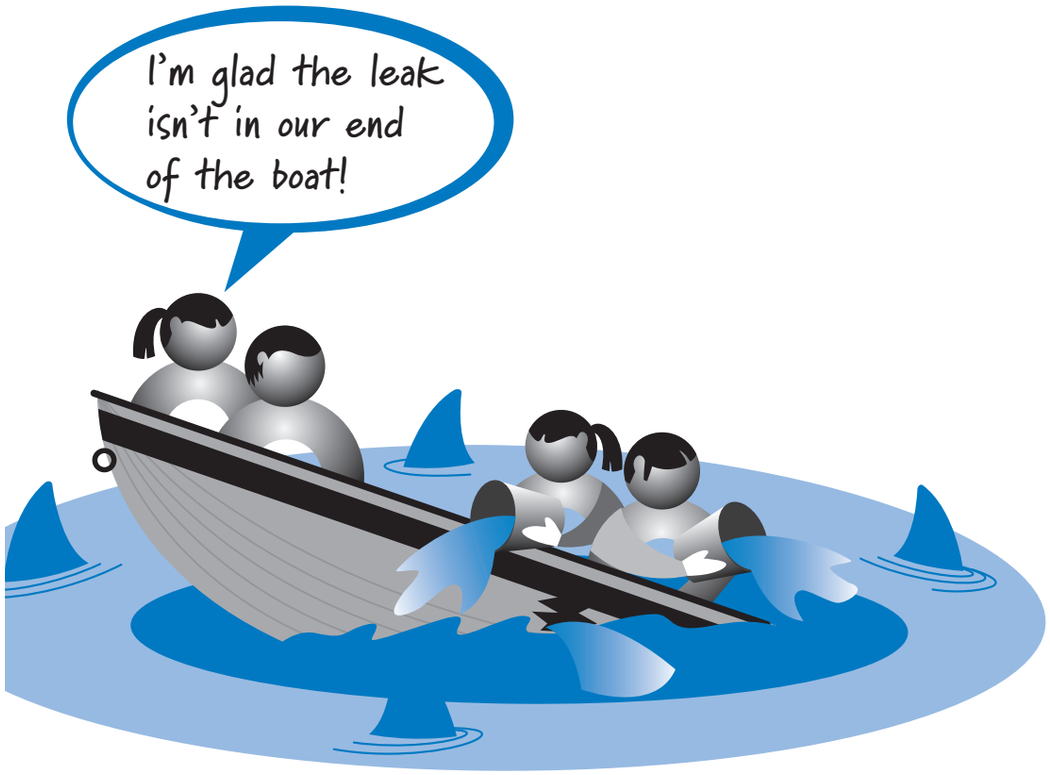



I ADDED ALL OF THE PRODUCT FEATURES THAT EACH OF YOU DEMANDED.
PRODUCT
NOW OUR PRODUCT IS A WORTHLESS HODGEPODGE OF COMPLEXITY.
I APPRECIATE YOUR INPUT. I COULDN'T HAVE FAILED WITHOUT YOU.
TEAM-WORK!
DilbertCartoonist@gmail.com
Dilbert.com
© 2013 Scott Adams, Inc./Dist. by Universal Uclick
2-25-13



Team work

- "All for one and one for all."
- "I got my part done. You didn't get your part done. Therefore we failed." This attitude misses the point that team members are all in the same boat together
- team members must appreciate that they must work together to meet commitments, because if they fail, it's going to be everybody's problem in the end
- Even if skills limitations prevent people from working cross-functionally, team members can still organise their work to ensure a good flow through the sprint



Performance

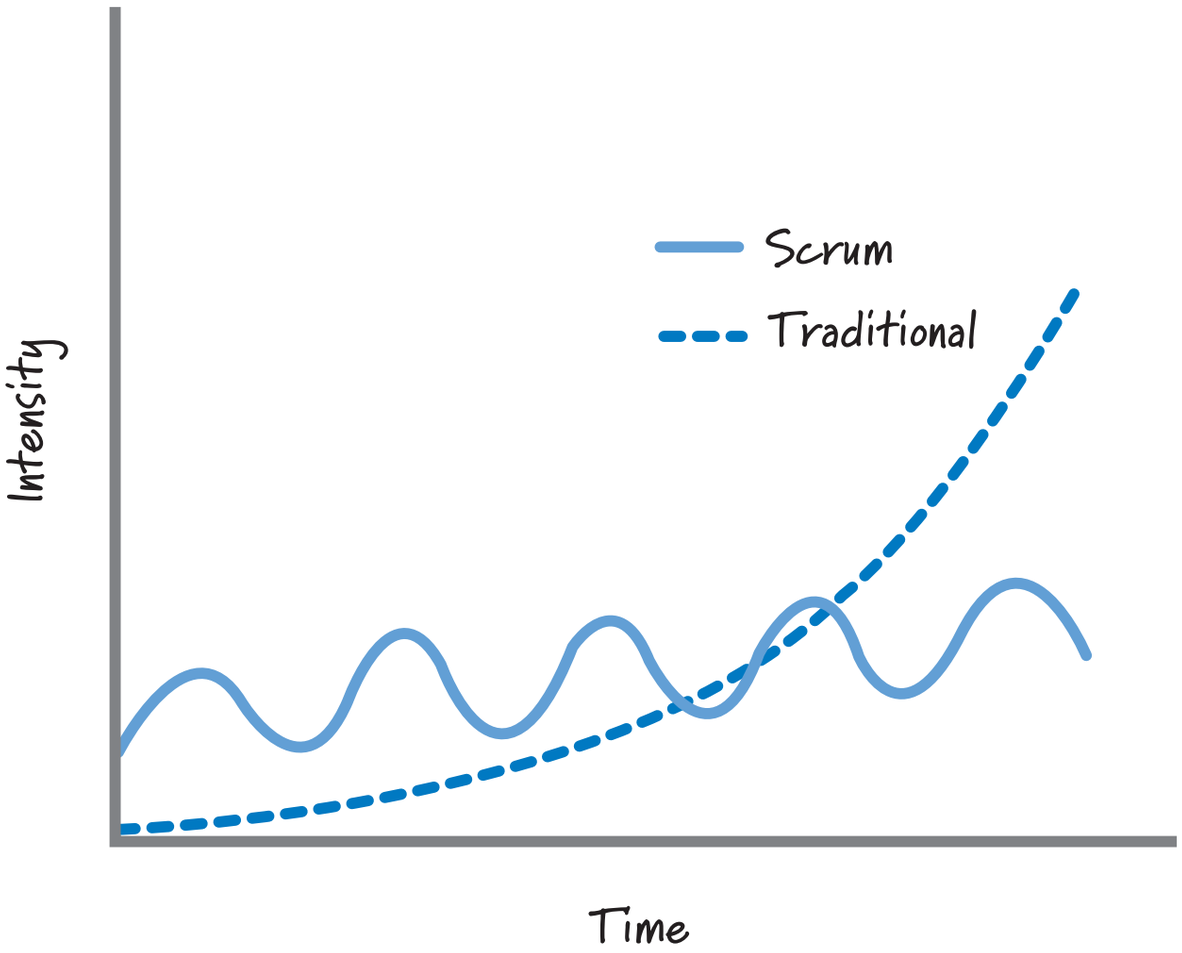
within a sprint we'll likely see intensity increase a bit near the end of the sprint

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



- **During the practical we can discuss your project plans**

Plagiarism Policy:

https://csiweb.ucd.ie/files/csi-plagiarism-policy_august2015.pdf

There is a lot of stuff online, and you must be careful that what you submit is your own work