Requirements Engineering

(Summer 2021)

Prof. Nan Niu (nan.niu@uc.edu)

https://github.com/nanniu/RE-Summer2021

© 2021, Nan Niu

Today's Menu

Monday (July 26)
RE Story Presentations

Tuesday (July 27):
Course Summary

Wednesday (July 28):
ASN4 Grading

The End!

© 2021, Nan Niu

Let's play a game

- Scope: All you need to know about RE (Summer 2021)
- Go to https://kahoot.it (ideally from a *different* device than where you launched Webex)
- Click "Play" (may be on top of the page)
- Enter the "Game PIN"
- Use *YOUR FULL NAME* to join the game

- Game rules:
  - Read each question via Webex
  - Select one and only one correct answer in *Kahoot!* within 30 seconds per question

© 2021, Nan Niu

What does it take to be an expert?

A person needs to know about 50,000 chunks of information to be an expert in a field, where a chunk is any piece of knowledge that can be remembered rather than derived.

Steve McConnell

Body of Knowledge

Source: *http://www.stevemcconnell.com/psd/05-BodyOfKnowledge.pdf*

Where did we start with?

→Requirements = stakeholder needs & desires

→Requirements are important because

- It's hard (*hardest*) to get them right.
- It's common to get them wrong.
- Getting them wrong is costly
- *Doing RE right saves money.*

© 2021, Nan Niu

The meaning of requirements

P, C |- S

E, S |- R

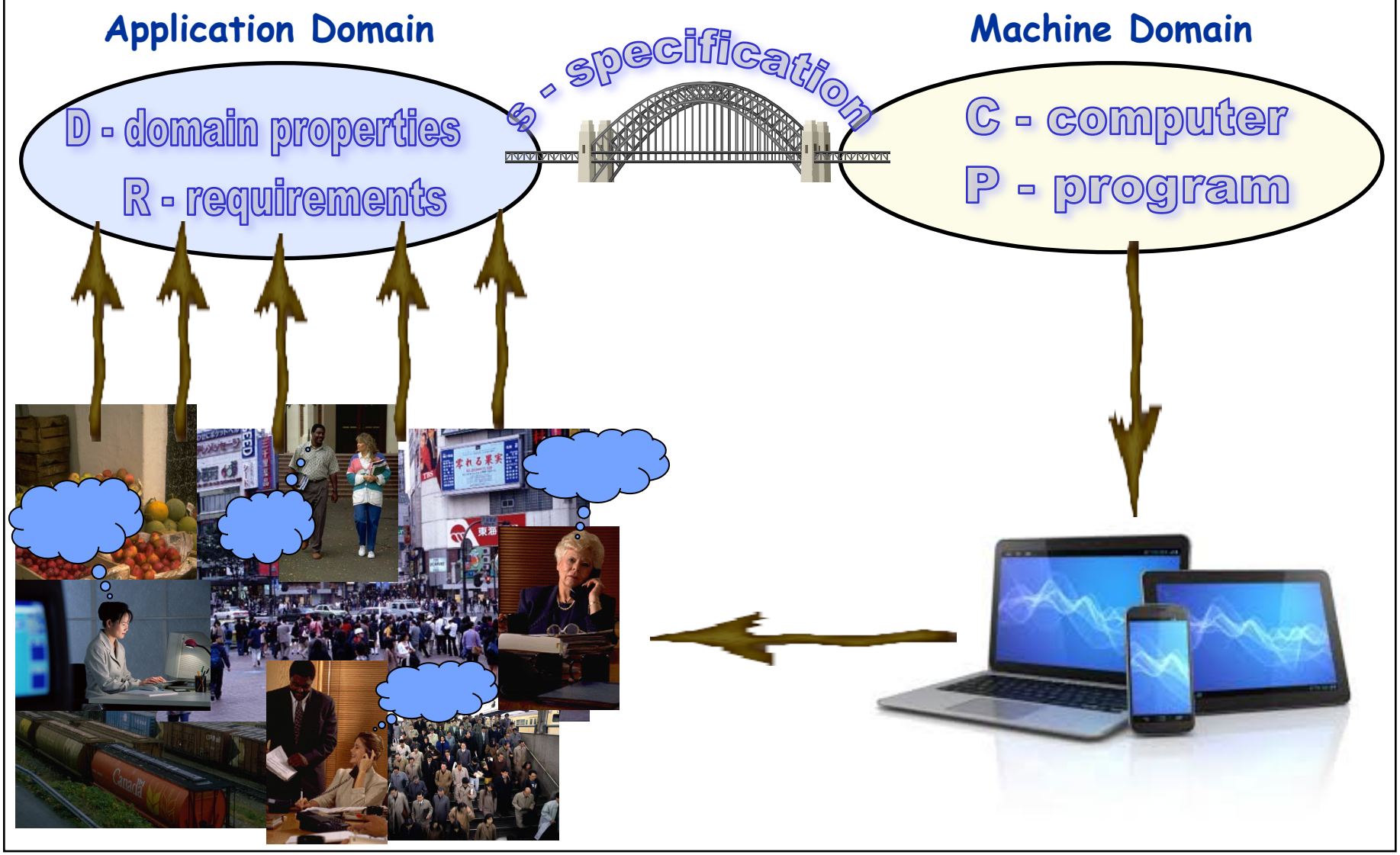
("|-" means entailment)

RE is more concerned with this one

Requirements engineers are an agent of change.

© 2021, Nan Niu

Requirements Engineer: Agent of Change



© 2021, Nan Niu

Req.s Elicitation vs. Req.s Gathering?

→Requirements ≠ What the customer said

→Requirements ≠

© 2021, Nan Niu

Req.s Elicitation vs. Req.s Gathering?

→ Requirements elicitation ≠ "asking the right questions"

Because there's (1) no right question to ask, (2) no right stakeholder to ask the question, and (3) no right answer.

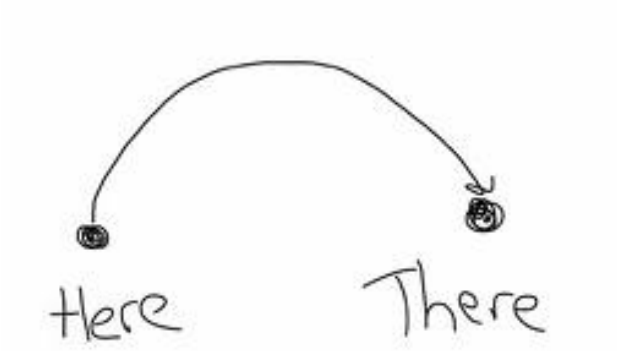
© 2021, Nan Niu

Elicitation Techniques

- Traditional techniques
  - Introspection
  - Reading existing documents
  - Analyzing hard data
  - Interviews
    - Open-ended
    - Structured
  - Surveys / Questionnaires
  - Meetings
- Collaborative techniques
  - Group techniques
    - Focus Groups
    - Brainstorming
  - JAD/RAD workshops
  - Prototyping
  - Participatory Design
- Cognitive techniques
  - Task Analysis
  - Protocol Analysis
  - Knowledge Acquisition Techniques
    - Card Sorting
    - Laddering
    - Repertory Grids
    - Proximity Scaling Techniques
- Contextual approaches
  - Ethnographic Techniques
    - Participant Observation
    - Ethnomethodology
  - Discourse Analysis
    - Conversation Analysis
    - Speech Act Analysis
  - Socio-technical Methods
    - Soft Systems Analysis

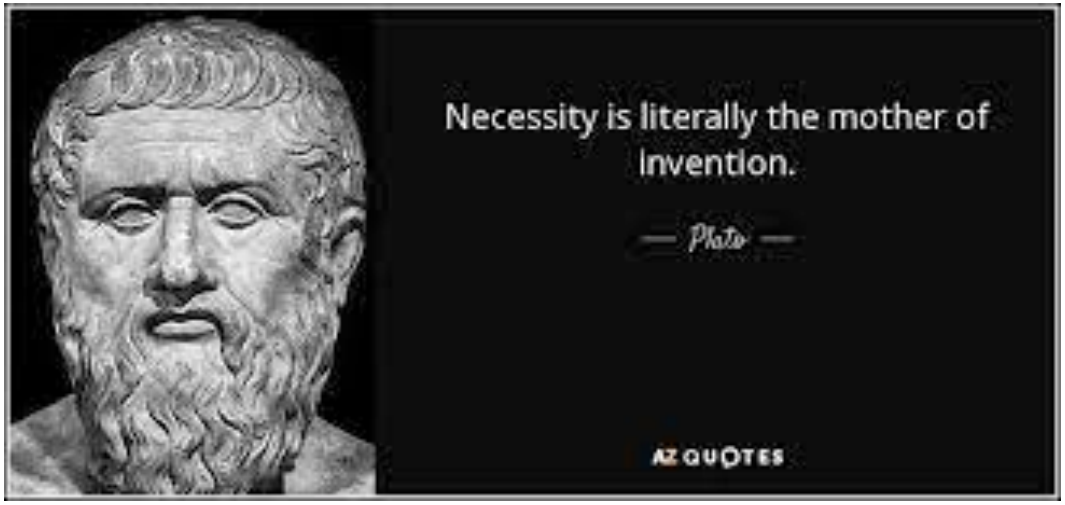
© 2021, Nan Niu

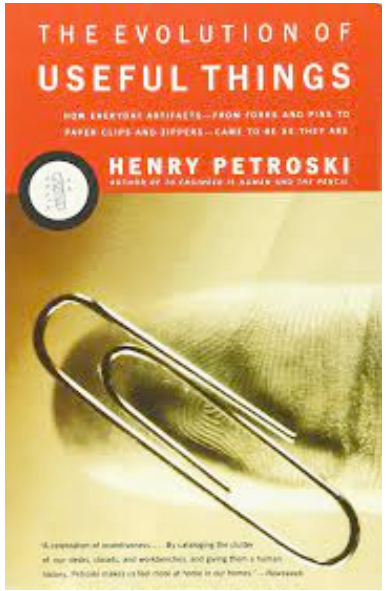
What does requirement come from?



What's perceived

What's desired

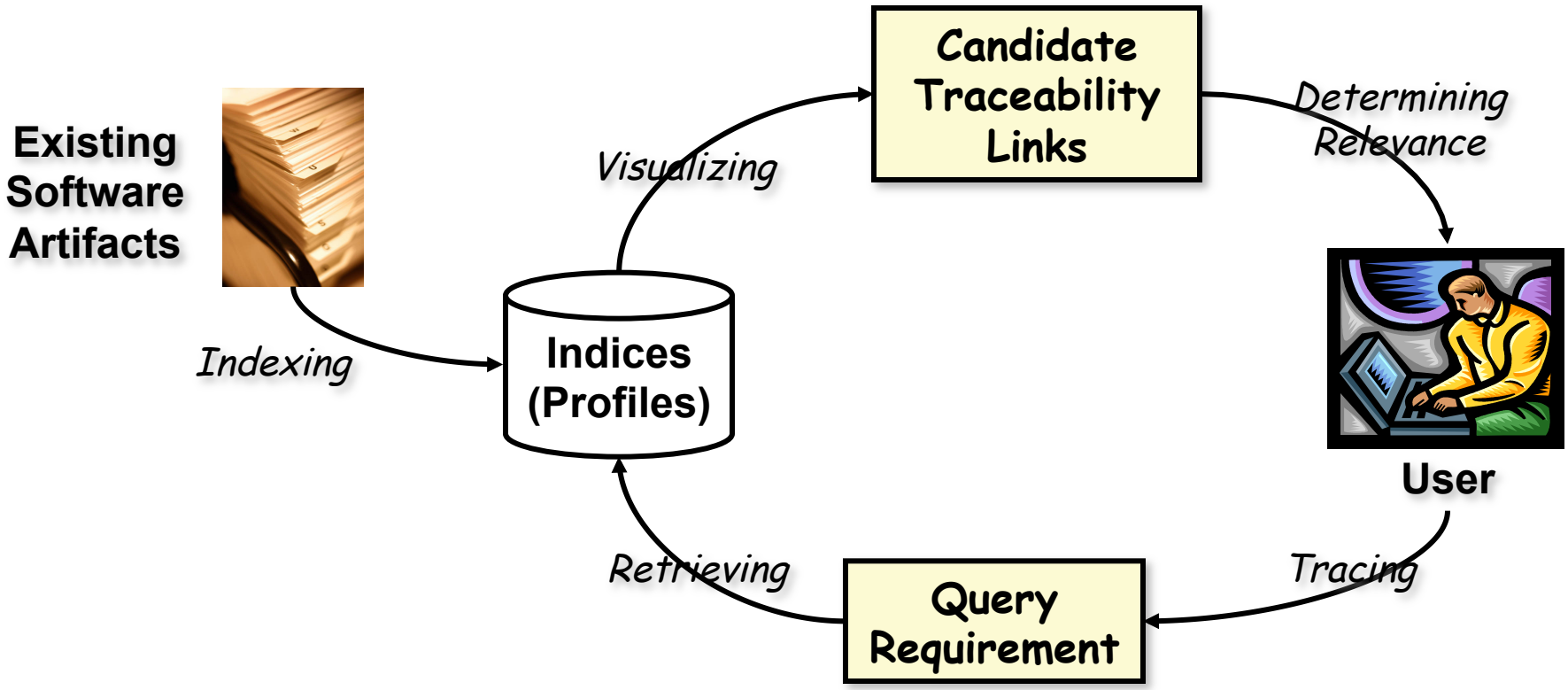




© 2021, Nan Niu

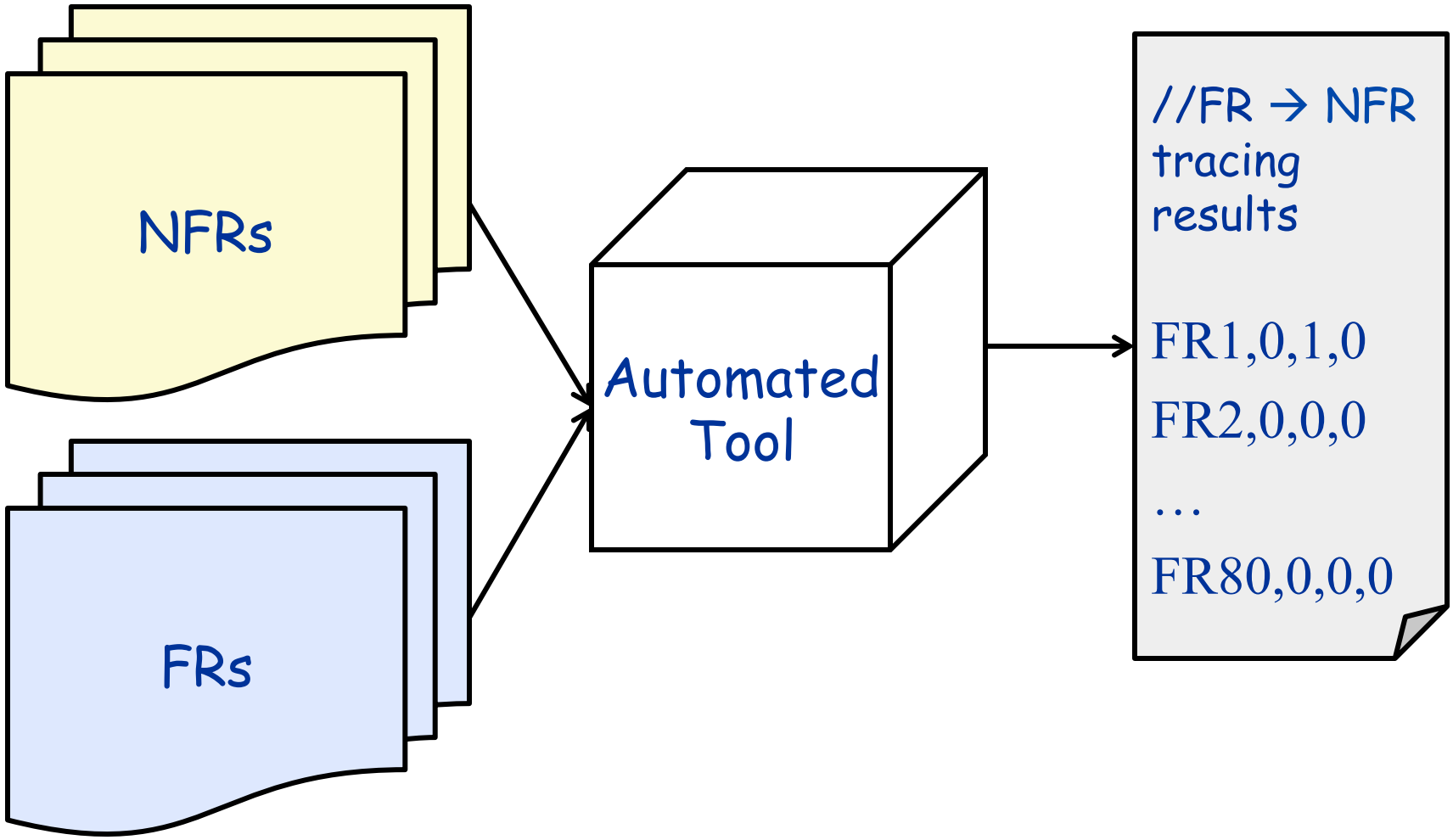
Source: *Google Images*

Requirements don't (and shouldn't) live independently



© 2021, Nan Niu

Linking NFRs and FRs



© 2021, Nan Niu

Grading Your ASN4 Solution

- Your ASN4 solution will be run three times on Wednesday (July 28)
  - Run #1: 80 FRs and 3 NFRs
  - Run #2: 100 FRs and 3 NFRs
  - Run #3: 100 FRs and 4 NFRs
- Your ASN4 solution shall write the three outputs into three different .txt files
  - YourName_Run1_Output.txt
  - YourName_Run2_Output.txt
  - YourName_Run3_Output.txt
- You will be asked to email the three output files to the instructor for grading

© 2021, Nan Niu

ASN4 Grading

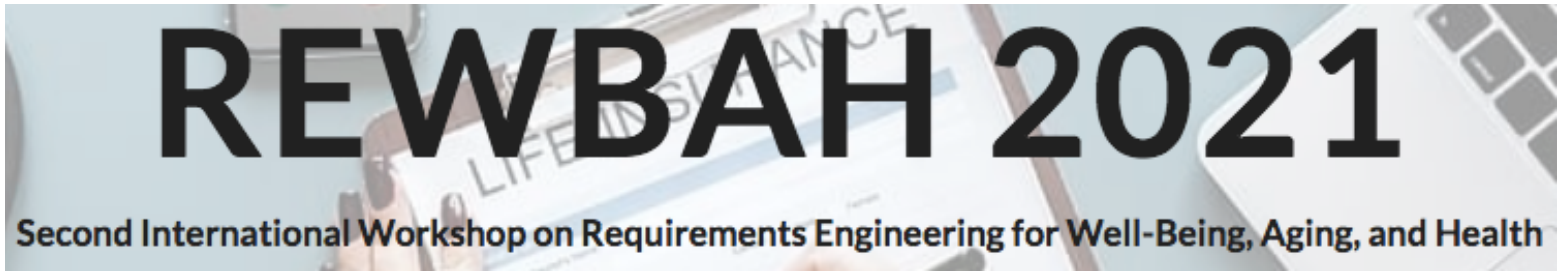
- If you belong to the odd number group {1, 3, 5, …, 17}, then please join Wednesday's grading session at ***9am***
- If you belong to the even number group {2, 4, 6, …, 18}, then please wait for the WeChat group message to join the class link on Wednesday. Note that your grading session will begin ***10am or later*** on Wednesday

1. Jiachang
2. Weijiang
3. Bo Z.
4. Bo L.
5. Jinzhi
6. Hongrong
7. Xiaoyu
8. Miaoyu
9. Shuang
10. Muyu
11. Luyao
12. Yuqi
13. Xiling
14. Xiaye
15. Chenxi
16. Zimao
17. Zichun
18. Pengxiang

© 2021, Nan Niu

RE: What's hot and what's not?

→ https://conf.researchr.org/track/RE-2021/RE-2021-workshops#Accepted-Workshops



© 2021, Nan Niu

Isn't "correctness" the ML req.?

"Machine learning is the science (and art) of programming computers so they can learn from data," writes Aurélien Géron in *Hands-on Machine Learning with Scikit-Learn and TensorFlow*.

Cat

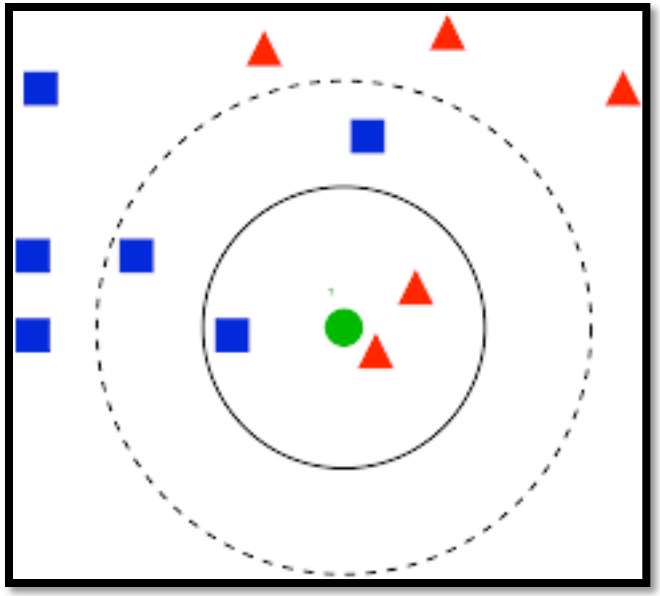
Dog

Source: *https://www.oreilly.com/content/machine-learning-a-quick-and-simple-definition/* and *Google Images*

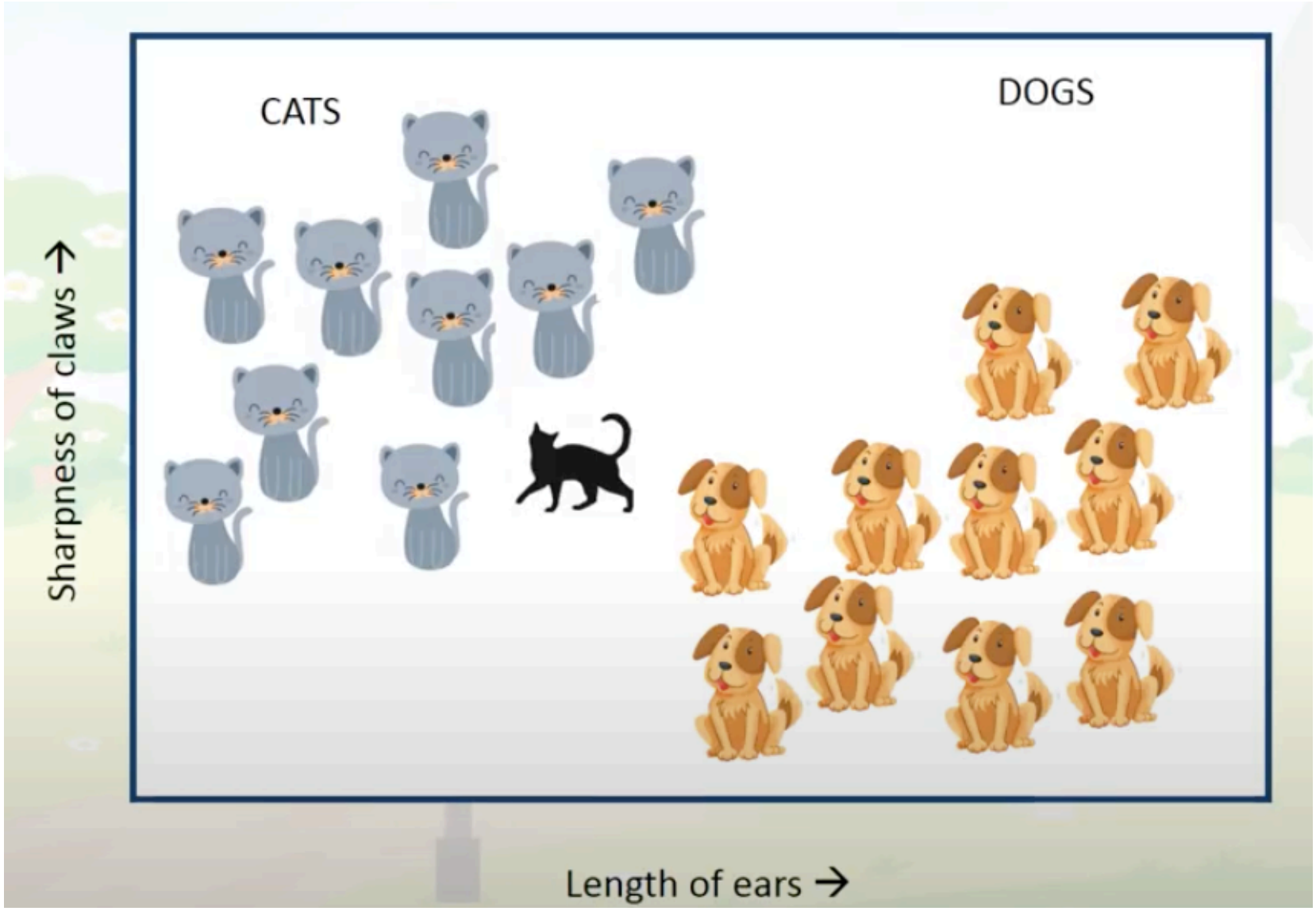
Let's consider ML via *k*-NN

→ *k*-NN (*k*-nearest neighbors)

- A non-parametric method: *k*>0 (typically small)
- Input: *k* closest training examples in the feature space
- Output: assign the to-be-classified object to the class most common among its *k* nearest neighbors

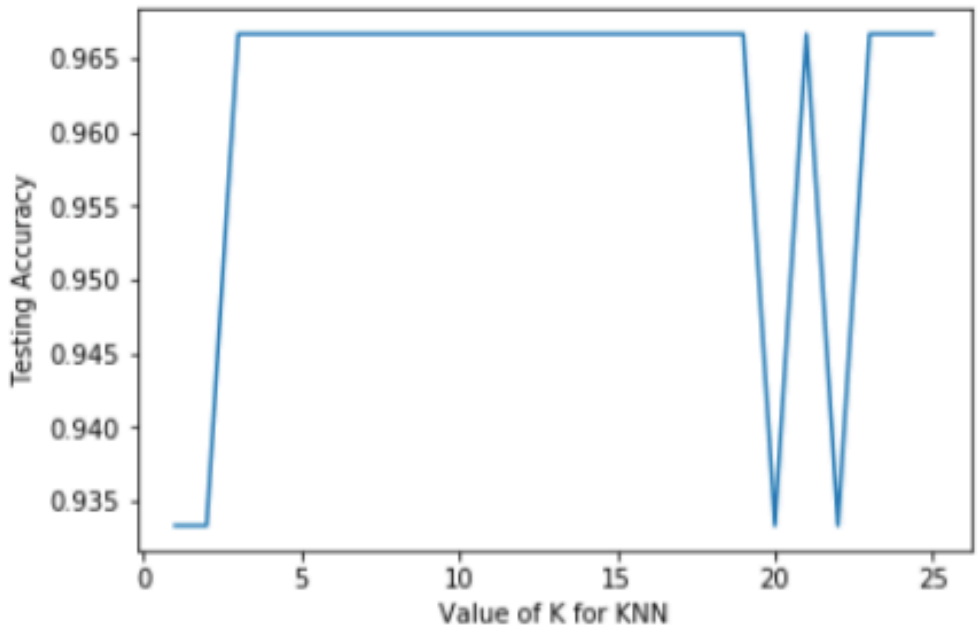


Source: *Wikipedia*



Source: https://www.simplilearn.com/tutorials/machine-learning-tutorial/knn-in-python

Learning *k*



For our final model we can choose a optimal value of K as 5 (which falls between 3 and 19) and retrain the model with all the available data. And that will be our final model which is ready to make predictions.

Source: *https://towardsdatascience.com/knn-using-scikit-learn-c6bed765be75*

Final, deployed ML code

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X,y)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

```
#0 = setosa, 1=versicolor, 2=virginica
classes = {0:'setosa',1:'versicolor',2:'virginica'}

#Making prediction on some unseen data
#predict for the below two random observations
x_new = [[3,4,5,2],
         [5,4,2,2]]
y_predict = knn.predict(x_new)

print(classes[y_predict[0]])
print(classes[y_predict[1]])
```

© 2021, Nan Niu

Source: https://towardsdatascience.com/knn-using-scikit-learn-c6bed765be75

Final, deployed ML code

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X,y)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

```
#0 = setosa, 1=versicolor, 2=virginica
classes = {0:'setosa',1:'versicolor',2:'virginica'}

#Making prediction on the unseen data
#predict for the below two random observations
x_new = [[3,4,5,2],
         [5,4,2,2]]
y_predict = knn.predict(x_new)

print(classes[y_predict[0]])
print(classes[y_predict[1]])
```

Is this ML code correct, if being correct is the req.?

Source: https://towardsdatascience.com/knn-using-scikit-learn-c6bed765be75

"Correctness" is w.r.t. the specification

→ *k*-NN (*k*-nearest neighbors)

- A non-parametric method: *k*>0 (typically small)
- Input: *k* closest training examples in the feature space
- Output: assign the to-be-classified object to the class most common among its *k* nearest neighbors

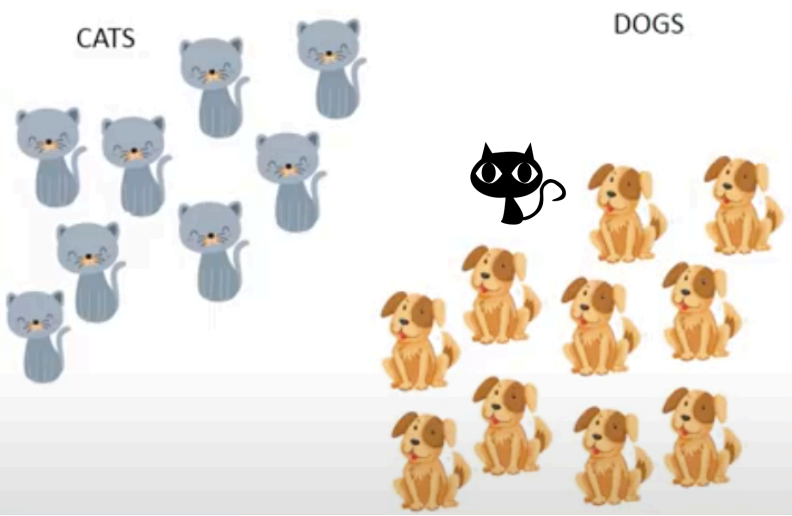
→ The above specifies the *k*-NN algorithm's design, and if the code *correctly* implements the design, then the code is *correct*.

© 2021, Nan Niu

But…

→… when classification errors occur, what (and who) to blame?

↳where the req.s really reside



→Type to "everyone" in Webex of what you think

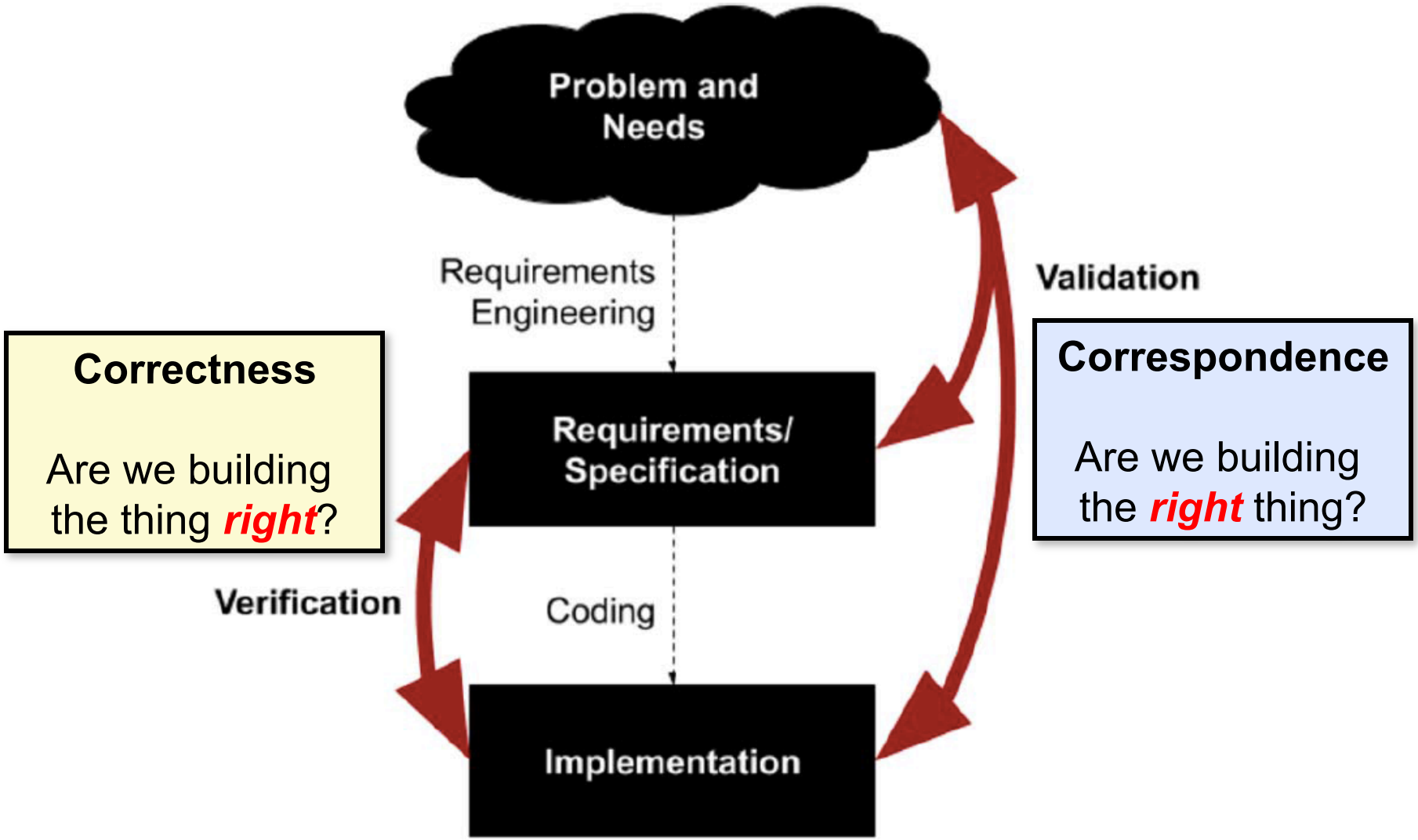
© 2021, Nan Niu

Most Popular: k-NN algorithm

Insufficient training	6
k-NN algorithm	8
Bad data	4
Bad requirements	6
Nothing to blame	4
Bad features	3
Time & budget	2
ML library	2

© 2021, Nan Niu

k-NN (the ML model) doesn't fit

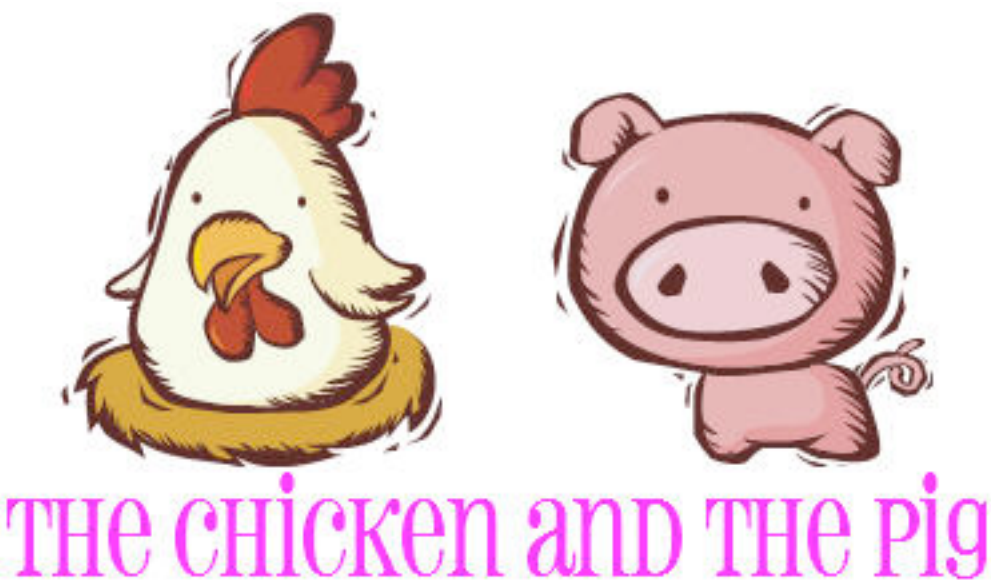


© 2021, Nan Niu

Final Remark

→ You're not only building the things right, but also building the right (better: more intelligent, more secure, more sustainability, cheaper, faster …) things!

→ My favorite RE Story



Source: *Google Images*

Requirements Engineering

(Summer 2021)

Great having you & keep in touch!

© 2021, Nan Niu