## 1. A

warehouse operations

SAAS

existing application in production

system never fails

normal operations module = m1+m2 are available

emergency operations module = m1+m2 are available

probalitiy of normal and emergency

probalitiy of normal functioning of warehouse operations module

minimum - 0.999, 0.998

answer

normal = m1\*m2

emergency = m3\*m4

normal = 0.995 x 0.998 = 0.99301 = 99.301%

emergency = .9995 x .9999 = 99.94%

parallel modules = 1 - ((1-0.99301) x (1-0.9994))

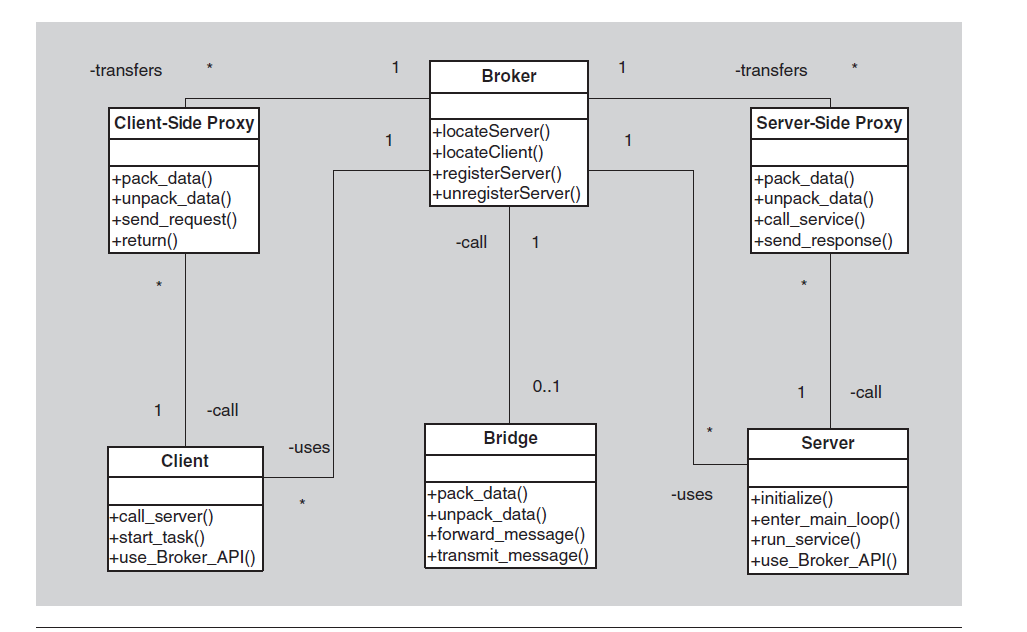
= 1-(0.00699 x 0.0006)

= 1-0.000004194

= 0.999995806

overall warehouse availability = operation\_strategy\_selector x parallel = 1 x 0.999995806 = 0.999995806

## 1.B



Brokers add a layer of indirection, and hence latency, between clients

and servers, and that layer may be a communication bottleneck.

The broker can be a single point of failure.

A broker adds up-front complexity.

A broker may be a target for security attacks.

A broker may be difficult to test.

## 1.C

* Availability
* Usability
* Adaptability
* Modifiability

## 1.B.C

2 application - operational failure char per day

application 1 crashes 10times - 5min to restart

application 2 crashes 20times - 2min to restart

Mean Time to Failure (MTTF)

– average time between observed failures (aka MTBF)

• Availability = MTTF / (MTTF+MTTR)

= (5\*10)/((5\*10)+5)

= 50/55

= 0.909

– MTTF = Mean Time To Failure

– MTTR = Mean Time to Repair

• Reliability = MTBF / (1+MTBF)

app1 :

24\*60 =1440

1440-10(times)

uptime = 1440 - 50 = 1390

availability = uptime / (uptime + downtime)

= 1390 / 1440 = 96.5277%

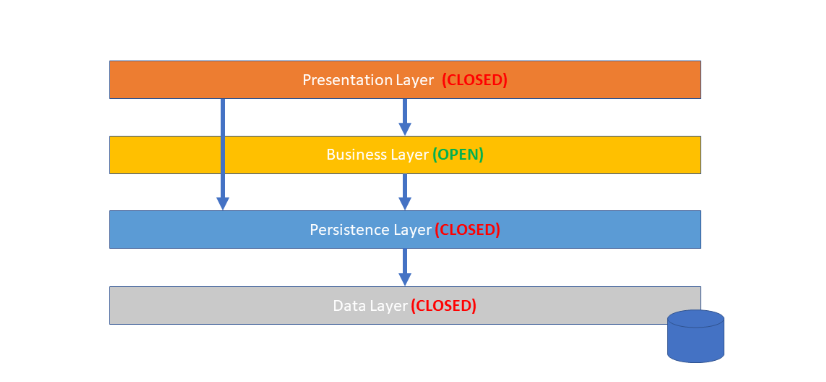
app2 :

uptime = 1440 - 40 = 1400

availability = uptime / (uptime + downtime)

= 1400 / 1440 = 97.2222%

## 2.A.a



## 2.A.b

## 2.A.c

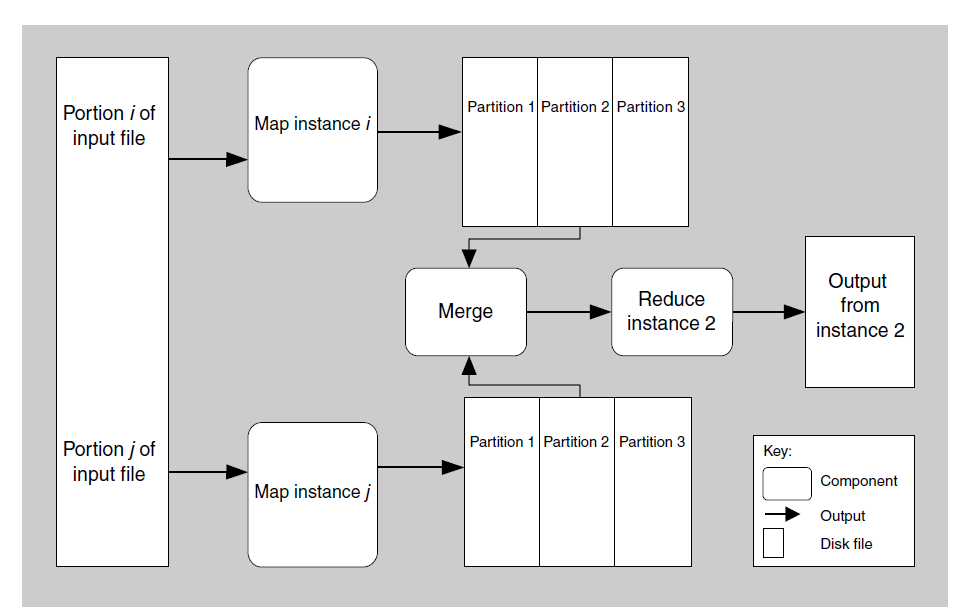
Creating unit testing for each layer with mocking

## 3.A

<https://www.techtarget.com/searchcio/definition/SMAC-social-mobile-analytics-and-cloud>

4.A.a

Map-Reduce Pattern



HDFS and Map reducer