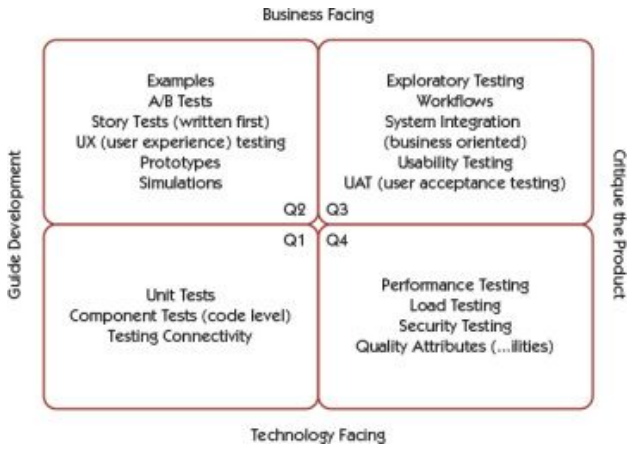
System Testing Study Point Exercises

The 4 agile testing quadrants
The Quadrants bliver brugt til at beskrive typer af test som der bliver brugt i Extreme Programming.
Der er ikke en bestemt rækkefølge, så man skal nødvendigvis ikke gå fra Q1 til Q4,
De hedder Q1-Q4 så vi kan kalde dem det i stedet for f eks.
"technology-facing tests that guide development" når vi snakker om dem.
Den venstre side af the Quadrants er om at sikre sig bugs før og imens man udvikler.
Den Højre side af the Quadrants er til for at finde bugs og finde "missing features".
Toppen af the Quadrants er test som hengiver sig hovedsageligt til "kunden/virksomheden".
Bunden af the Quadrants er til udviklings teamet, men samtidig har en stor faktor i at få et succesfuldt program.

De fire Quadrants er:
Q1: technology-facing tests that guide development
Q2: business-facing tests that guide development
Q3: business-facing tests that critique (evaluate) the product
Q4: technology-facing tests that critique (evaluate) the product



Q1
Indeholder f eks. Unit tests og Component tests, det er tests som kan køre automatisk.
I Q1 er målet, at man sikre sig at lave kvalitet kode.

Q2
Henvender sig til kunden, f eks. prototyper, hvor man lave en prototype af noget, hvorefter kunden kan give feedback.
Q2 er nødvendigvis ikke noget man kan benytter i starten af udviklingen, den kan være effektive igennem hele udviklingsfasen.

Q3
Målet her er at sikre sig at brugeren af systemet, kan finde ud af at benytte systemet, det gør man ved at demo det med kunden/brugeren.

Q4
Her tester man om systemet kan klare at f eks. 10000 brugere benytter systemet og sender requests frem og tilbage, hvis det feks er en hjemmeside, så kan det være at hjemmesiden går ned, eller hvis der er en database tilknyttet, så kan det være at den ikke kan holde til det.

System testing
Er at teste et komplet integreret system, hvor man måler op imod de requirements som der beskriver systemet, så når man benytter denne test, så er det vigtigt at kigge på hele systemet som en helhed og ikke teste det som mange forskellige komponenter.
System testing hører til under “black-box testing”, og man behøver ingen forståelse for infrastrukturen som består af design og kode, for at kunne lave en system test.

Exploratory Testing
Er en teknik som udfordrer dig til at tænke anderledes omkring din test, og derfor kan du til tider finde tests cases som du ikke havde tænkt over da du planlagde din test, og det kan hjælpe dig til at få et mere kvalitets fuldt system.
Man tester en GUI og prøver at skrive forskellige tegn,sætninger, typer(f eks. boolean,int,string), for at se om systemet har sikret sig imod fejlagtige inputs.

Jmeter
10 users:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec	Avg. Bytes
HTTP Request	10	45	40	52	4,61	0,00%	1,1/sec	0,34	0,20	312,5
TOTAL	10	45	40	52	4,61	0,00%	1,1/sec	0,34	0,20	312,5

100 users:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec	Avg. Bytes
HTTP Request	100	72	34	3033	297,61	0,00%	10,1/sec	3,07	1,86	312,0
TOTAL	100	72	34	3033	297,61	0,00%	10,1/sec	3,07	1,86	312,0

1000 users:

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec	Avg. Bytes
HTTP Request	1000	11494	36	21295	9032,84	44,40%	31,1/sec	34,99	3,20	1153,0
TOTAL	1000	11494	36	21295	9032,84	44,40%	31,1/sec	34,99	3,20	1153,0

Det ser ud til at serveren ikke kan klare presset fra 1000 users, da fejlprocenten er steget med hele 44%, så hvis det ikke var en reel hjemmeside som skulle kunne håndtere mange brugere ville jeg flytte fra min gratis shift server over til noget der kan klare presset.