Testautomation

Varför, vad, när, vem och hur?

Claremont

ABOUT CLAREMONT

Management and IT consultancy firm that help companies on their digital journey.

Digital journey

Offering expertise ranging

- Digital strategy to solution
- Development
- Integration
- Service management

Mission

Empower and enable our customer's business throughout their digital transformation

Our responsibility

Combat poverty through enterpreneurship Kannankudi, in southeast India

Profitable business

- Revenue 226 Mkr
- Profitable 14%,
- Human Growth 30%

When a person grows, the company grows with that person

250 experienced consultants

- Project Management
- Process
- Quality Management
- Test Automation
- ERP System
- Architecture and Development
- Business Intelligence

Company culture

- Commitment
- Professionalism
- Business Driven

Partner

- Microsoft
- EpiServer
- InRiver
- CA Technologies
- HP

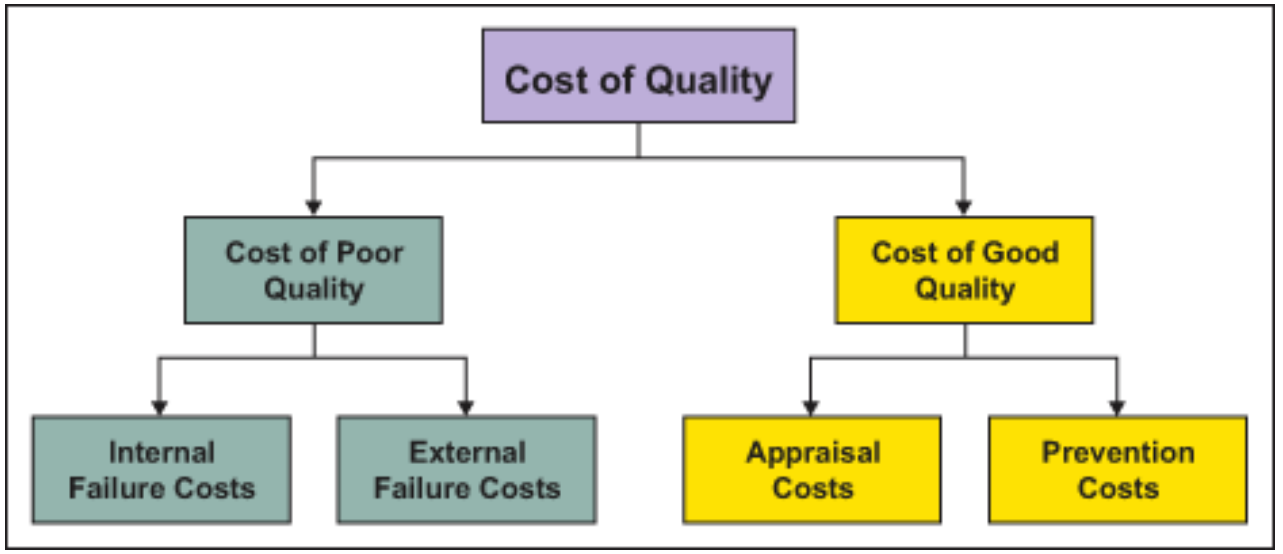
Price Awarded


Q ?

Q = 

Q = 

Cost of Quality - SixSigma



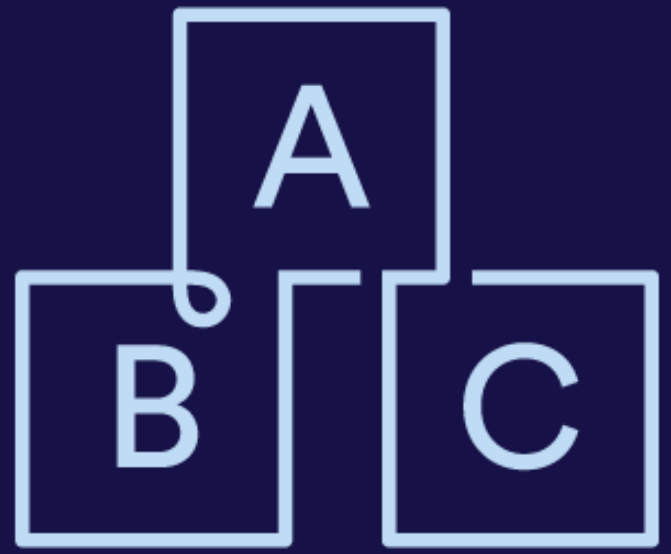
Q = subjektivt

Q = Situations-beroende

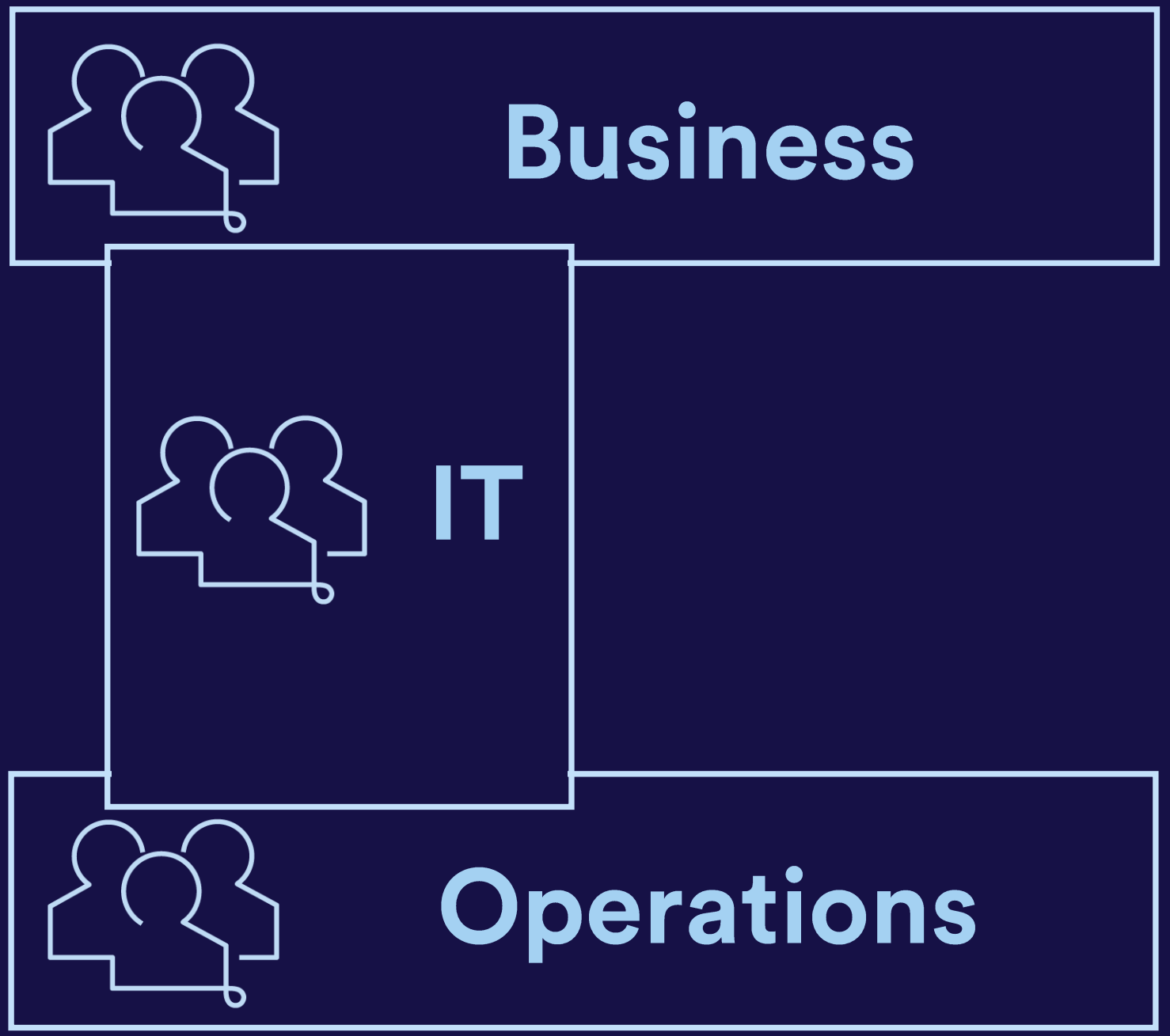
Business

IT

Operations

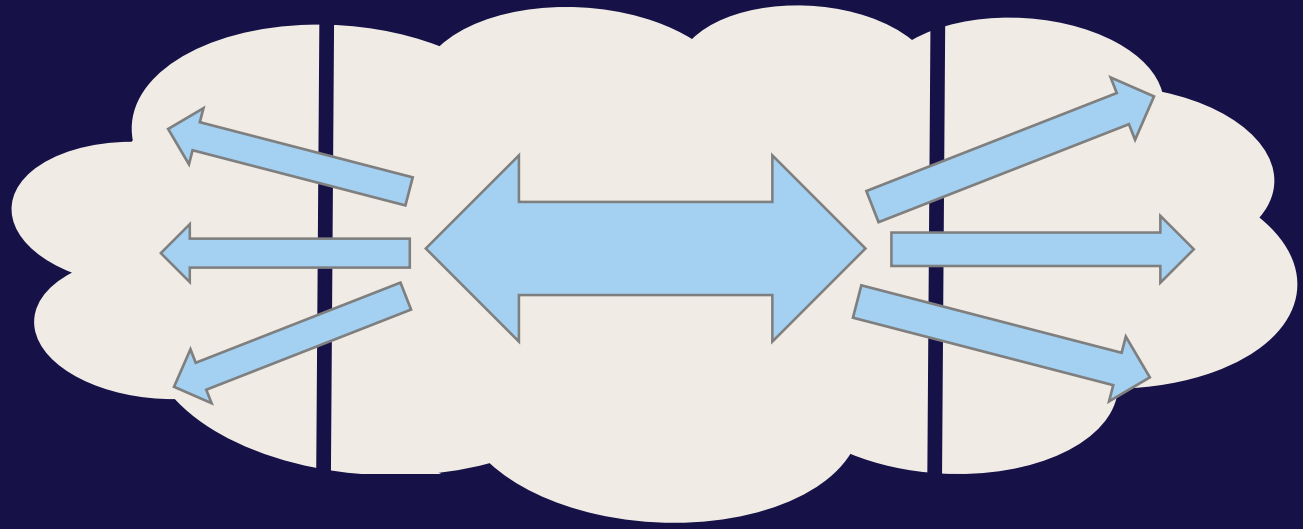


Agile

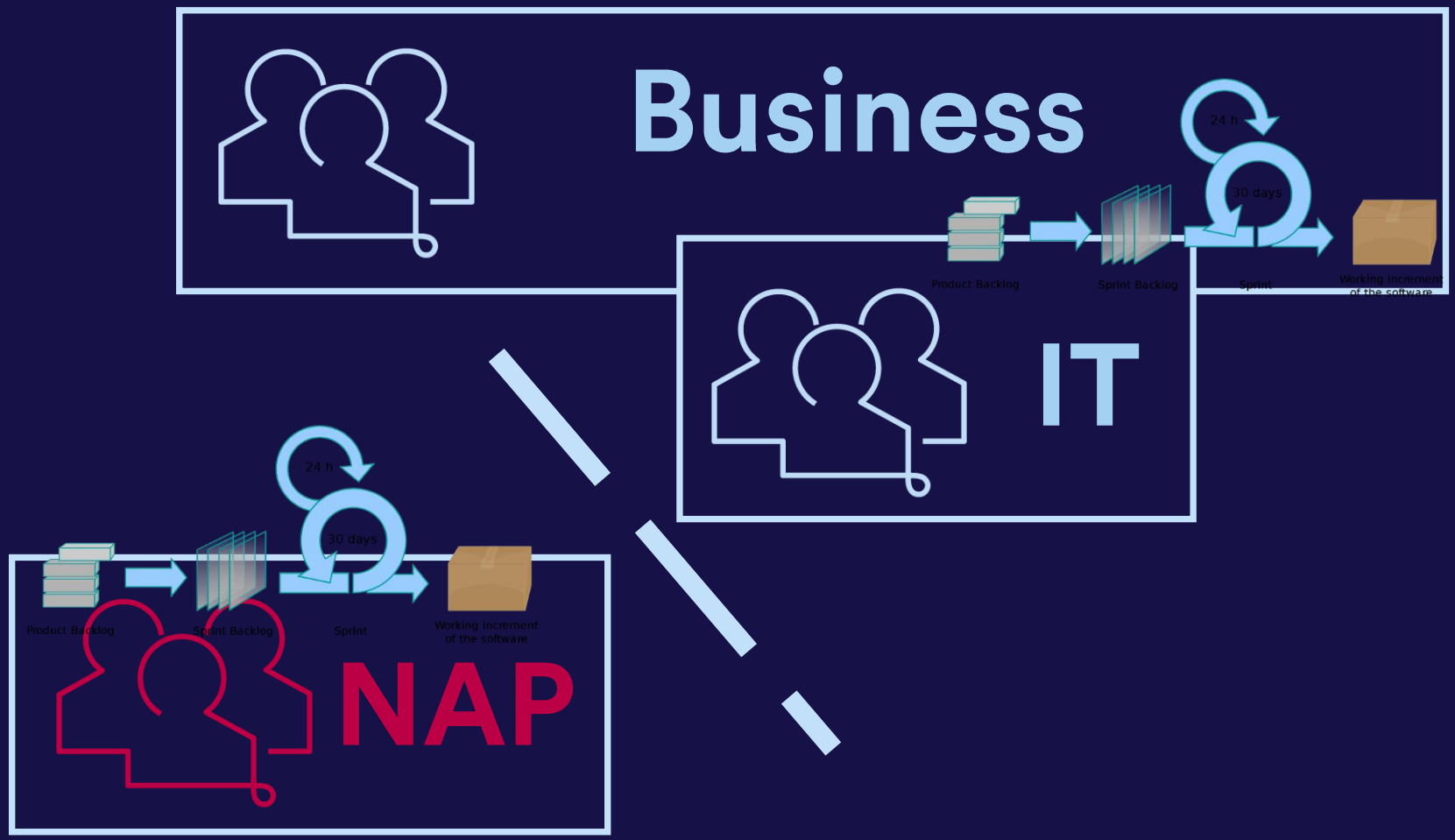


DevOps

Agile



NAP



NAP

Operations

Vad skapar tester för något?

Visshet

Nöjdhet

Smörjolja

Information

Kvalitetsegenskaper för programvara

Olika intressenter

Olika faser

Ledtider

Förmågor. Erbjuder produkten värdefulla funktioner?

- Komplett: alla viktiga funktioner som önskas av användare är tillgängliga.

- Korrekt: utdata eller beräkningar i produkten är korrekta och presenterade med signifikanta siffror.

- Effektiv: utför sina funktioner på ett effektivt sätt (utan att göra mer än vad som krävs).

- Interoperabilitet: olika funktioner i produkten fungerar med varandra på bästa sätt.

- Samtidighet: möjligheten att utföra många parallella uppgifter, t.ex. att kunna köra samtidigt som andra processer.

- Datastatistisk: [illegible] format och hanterar data korrekt.

- Utökningsbar: möjlighet för kunder eller tredjepartsleverantörer att lägga till egna funktioner eller förändra beteendet.

Pålitlig. Kan du lita på produkten i många och svåra situationer?

- Stabilitet: produkten ska inte krascha , orsaka undantag eller skriptfel.

- Robust: produkten hanterar (o)förutsedda fel på ett behagligt sätt.

- Stresstålighet: hur beter sig systemet när olika gränser överskrids.

- Återhämtning: det är möjligt att starta om och fortsätta efter ett allvarligt fel.

- Datakonsistens: all [illegible] data bevaras intakt genom hela produktens

- Säkerhet: produkten medverkar inte till att skada personer eller egendom.

- Katastrofhantering: [illegible] händer ordentligt, riktigt allvarligt inträffar.

- Tillförlitlighet: [illegible] produktens beteende konsekvent förutsägbart och trovärdigt.

Användbarhet. Är produkten lätt att använda?

- Inbjudan: produkten inbjuder till att upptäcka möjligheter i produkten.

- Intuitiv: det är lätt att förstå och förklara vad produkten kan göra.

- Minimalistisk: produktens innehåll eller utseende innehåller inget onödigt.

- Lättlärd: det går snabbt och lätt att lära sig hur produkten ska användas.

- Ihågkombarhet: har du lärt dig hur du ska utföra något, glömmer du inte bort det.

- Utforskningsbar: information och möjligheter om produkten kan upptäckas genom att utforska användargränssnitt

- Snabbanvänd: [illegible] kan utföra vanliga uppgifter väldigt snabbt

- [illegible] har [illegible] och utmärkt [illegible] interaktion via användargränssnitt

- Kontroll: användarens känsla [illegible] kontroll över programvaran genom hela användarscenarier

- Klarhet: är allting explicit och [illegible] uttryckt med ett [illegible] som kan bli förvirrande och intetsägande?

- Felhantering: [illegible] informativa, [illegible] svårt [illegible] misstag och att reparera

- Konsekvent: beteenden är samma genom hela produkten, och det är igenkännligt utseende och känsla.

- Anpassningsbart: förvalda inställningar och beteenden kan specificeras för att uppnå flexibilitet.

- Tillgänglighet: produkten kan användas av så många människor som möjligt och möter aktuella tillgänglighetsriktlinjer.

- Dokumentation: det finns en Hjälp som hjälper och som beskriver den aktuella funktionaliteten.

Karisma. Har produkten "det"?

- Unikhet: produkten är säregen och har någonting som andra inte har.

- Tillfredsställelse: hur känner du dig efter att ha använt produkten?

- Professionalism: andas produkten rätt typ av professionalism och känns den anpassad för syftet?

- Attraktivitet: är alla aspekter av produkten estetiskt tilltalande för ögon och andra sinnen?

- Väcker nyfikenhet: kommer användarna bli intresserade att prova sig fram i produkten?

- Uppslukande: blir användarna fast, satta i trans, och helt uppslukade när de använder produkten?

- Hajp: använder produkten för mycket eller för lite av den bästa och senaste tekniken och idéerna?

- Förvärvar [illegible] produkten överträffar förväntningar och [illegible] du inte visste att du ville ha?

- Attityd: har produkten och dess information den [illegible] attityd och pratar till dig med [illegible] språk och stil?

- Förstaintryck: (första) intryck [illegible] imponerande?

- Historier: finns det intressanta historier om produktens uppkomst, konstruktion eller användande?

Hur väl stoppar produkten [illegible] skada att [illegible]

- Autentisering: produktens igenkännande av användare.

- Auktorisering: produktens hantering av vad autentiserade användare kan se och göra.

- Privathet: förmågan att inte avslöja data som är skyddad mot icke-auktoriserade användare.

- Säkerhetsluckor: produkten ska varken erbjuda eller inbjuda till sårbarhetsintrång.

- Hemlighetsfull: produkten ska under inga omständigheter avslöja information om underliggande system.

- Osårbarhet: förmågan att motstå försök att penetrera säkerheten.

- Riskfri: produkten [illegible] inte [illegible] förebefordra [illegible] eller ge upphov till det av att vara ett.

- Kopieringsskydd: ingen [illegible] möjlighet att illegalt kopiera eller distribuera programvaran eller koden.

- Reglementsenlig: produkten uppfyller fastställda säkerhetsstandarder.

Uppfyllande av [illegible] produkten tillräckligt snabb?

- Kapacitet: produktens möjligheter/begränsningar, i många olika situationer (t.ex. slött nätverk)?

- Resursutnyttjande: välanpassat användande av processor, minne, lagring och andra resurser.

- Respons: (den upplevda) snabbheten när funktioner utförs.

- Upptid: systemet är tillgängligt för användande vid alla tidpunkter det ska vara.

- Genomströmning: produktens förmåga att processera många saker samtidigt.

- Uthållighet: kan produkten hantera last över lång tid?

- Återkoppling: svarar systemet tillräckligt snabbt när användare utför funktioner?

- Skalbarhet: hur bra skalar systemet upp, ner, eller ut?IT-vänlig. Är produkten lätt att installera, underhålla och supporta?

- Systemkrav: förmågan att [illegible] på stödda konfigurationer och att hantera olika miljöer eller saknade komponenter.

- Flyttbarhet: produkten kan installeras på avsedda plattformar och använder en rimlig storlek av [illegible] hårdvarukomponenter.

- Uppgradering: det är enkelt att uppgradera till en ny version utan att tappa konfigureringsdata och installationsvar.

- Avinstallation: all [illegible] (förutom användar- eller systemfiler) och andra resurser ska tas bort under en avinstallation.

- Konfiguration: kan installationen konfigureras på olika sätt eller ställen för att stödja kundens användande?

- Utrullningsbarhet: produkten kan rullas ut av IT-avdelningen till olika typer av (begränsade) användare och miljöer.

- Underhåll: är produkten och dess artefakter lätta att underhålla och supporta för kunderna?

- Testbarhet: hur effektivt kan den utrullade produkten bli testad av användarna?

Kompatibilitet. Hur väl interagerar produkten med andra programvaror och miljöer?

- Hårdvarukompatibilitet: produkten kan användas tillsammans med applicerbara konfigurationer av hårdvarukomponenter.

- Operativsystemkompatibilitet: produkten kan köra på avsedda versioner av operativsystemen och följer deras riktlinjer.

- Programkompatibilitet: produkten, och dess data, fungerar ihop med andra program som kunderna använder.

- Konfigurationskompatibilitet: produktens förmåga att smälta in med [illegible] konfigurationer.

- Bakåtkompatibilitet: [illegible] produkten [illegible] den förra versionen kunde?

- Framåtkompatibilitet: [illegible] produkter att kunna använda artefakter eller gränssnitt i framtida versioner.

- Hållbarhet: påverkan på miljön, t.ex. [illegible] energieffektivitet, besparingar, möjligheter till distansarbete.

- Reglementsenlig: [illegible] uppfyller [illegible] applicerbara standarder, regelverk, lagar, riktlinjer eller förhållningssätt.

Support. Kan kundernas användning och problem understödjas?

- Identifierbar: det är lätt att identifiera delar av produkten och deras version, eller specifika fel.

- Diagnostisering: är det möjligt att få reda på detaljer kring kundsituationer?

- Undersökningsbar: är det lätt att sätta fingret på fel (t.ex. loggfil) och hjälpa?

- Avlusning: kan du observera programmets interna tillstånd när det behövs?

- Mångsidighet: möjligt att använda produkten på fler sätt än vad det designades för.

Testbarhet. Är det lätt att verifiera och testa produkten?

- Spårbar: produkten loggar händelser på lämplig nivå, och i användbart format.

- Styrbar: möjligheten att sätta tillstånd, objekt och variabler.

- Observerbar: möjligheten att observera det som testas.

- Övervakningsbar: kan produkten indikera hur och vad den gör?

- Isolerbar: möjligheten att testa en del för sig själv.

- Oföränderlig: förändringar till programvaran görs kontrollerat, och inte för ofta.

- Automation: finns det publika eller interna programmatiska gränssnitt som kan användas?

- Information: möjligheten för testare att lära sig det man behöver veta...

- Granskningsbar: kan produkten, och sättet den skapats på, valideras?

Underhåll. Kan produkten underhållas och utökas till låg kostnad?

- Flexibel: är det lätt att ändra produkten för att möta kunders behov?

- Utökningsbar: blir det lätt att lägga till funktionalitet i framtiden?

- Enkel: koden är inte mer komplex än vad som krävs, och försvårar inte testdesign, testekekvering och resultattolkning.

- Läsbar: koden är dokumenterad och lätt att läsa och förstå.

- Genomskinlig: Är det lätt att förstå de underliggande strukturerna?

- Modulär: koden är uppdelad i lätthanterliga bitar.

- Omskrivningsbar: är du nöjd med enhetstesterna?

- Analyserbar: möjligheten att hitta orsaker till problem, eller annan intressant kod.

Flyttbarhet. Är det möjligt att flytta produkten till andra miljöer och språk?

- Återanvändning: kan delar av produkten användas på andra ställen?

- Anpassningsbar: är det lätt att förändra produkten så den fungerar i en annan miljö?

- Kompatibel: följer produkten vanliga beteenden, eller officiella standards?

- Internationalisering: det är lätt att översätta produkten.

- Lokalisering: är alla delar av produkten anpassade för att möta behoven av det nya landet/kulturen?

- Robusta gränssnitt: kommer produkten se lika bra ut när den översatts

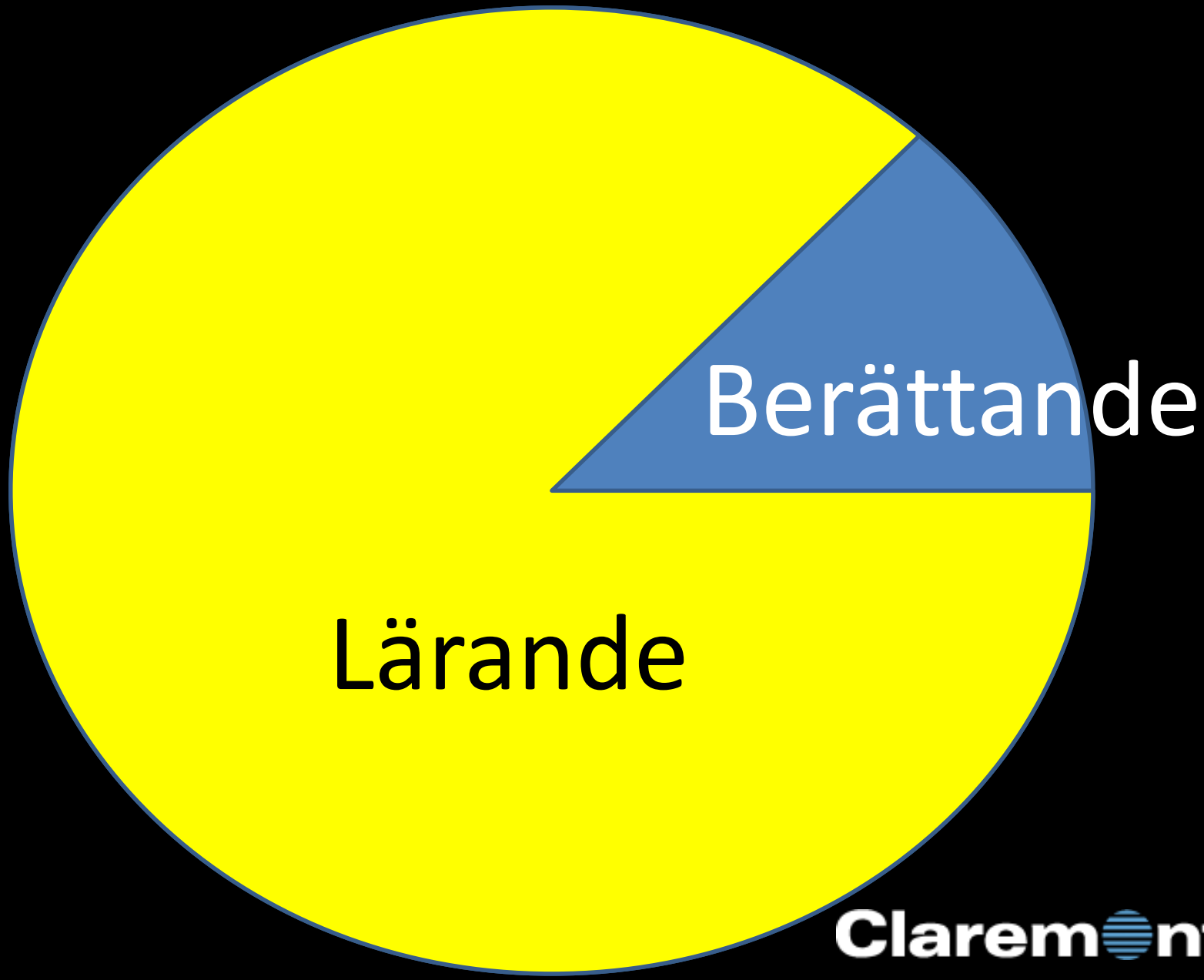
http://thetesteye.com/

Testningens verkliga uppgift

- Tänka på allt som ingen annan tänkt på
- Finna ut var olika människor förstått eller tolkat samma sak olika
- Upptäcka alla förutsättningar som ingen annan upptäckt
  - Specialfall i data eller användning
  - Användningsområden
  - Tidsberoenden
  - Tekniska beroenden
  - osv
- Påtala glömda aktiviteter
- Hitta var någon råkat gjort fel

Claremont

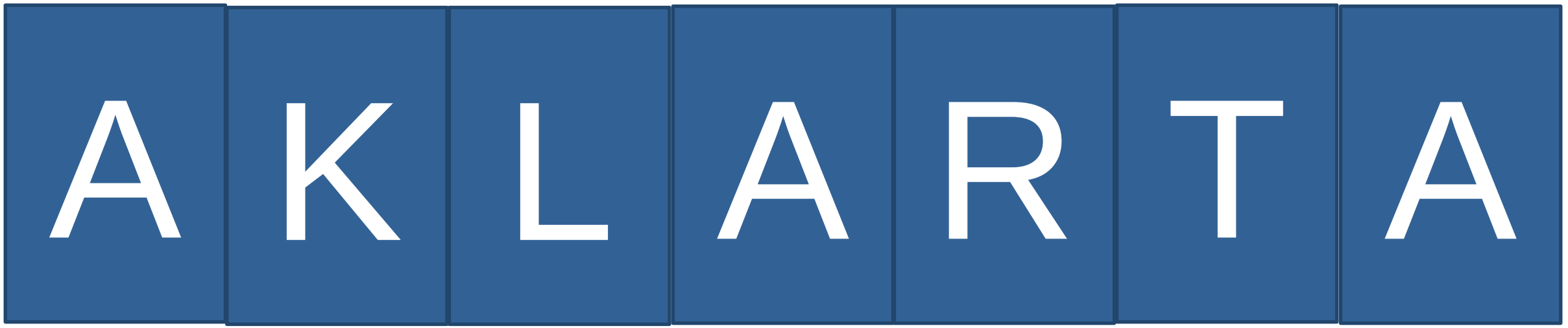
Krav- och testarbete



Claremont

Krav


12

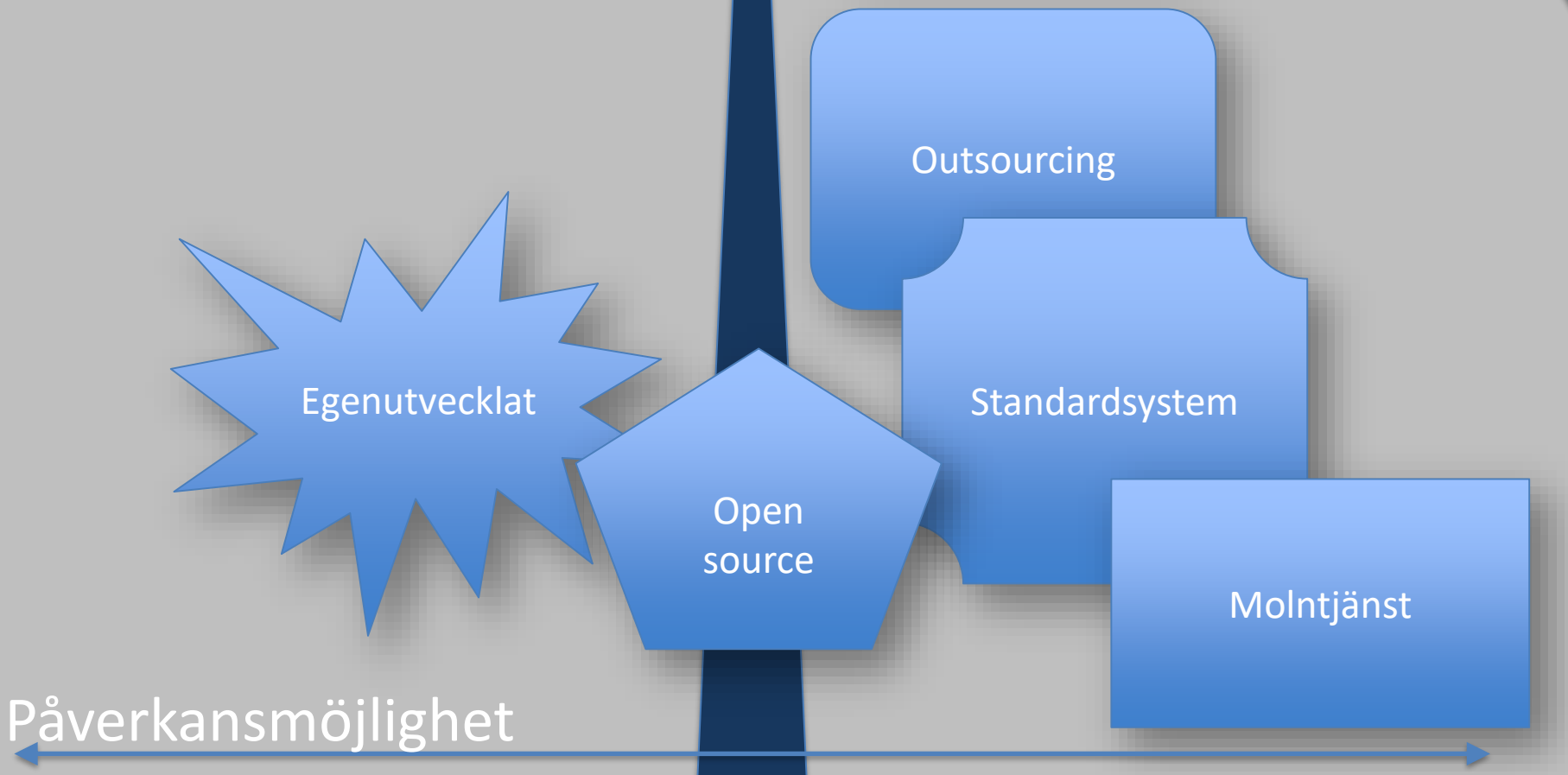


Kodat Levererat Accepterat Redovisat Testat

Avstämt Kodat Levererat Avstämt Redovisat Testat Accepterat

Nedbrytning (UT/IT)
Konstruerat data
Mockar
CI/CD
UX
ST

Avidentifiering
Sub-setting
Up-stream
Avtal
UAT
SV



[au-tu-ma-tiˊsé-rað]

Begrepp

Term	Förklaring
Regressionstest	Tester som görs för att säkra att ingenting gått sönder när man infört en ändring. Testar i hela systemet, inte bara runt ändrade funktionaliteten.
Omtest	Testfall som körs igen, t.ex. vid felrättning.
Utforskande testning	Tester där man dokumenterar testfallen medan man utför dem.
”Krav”	Innefattar allt som kommunicerar förväntningar på systemen. Utgörs vanligen av User Stories, Feature cards, atomära krav, användningsfall, dialogbeskrivningar o.s.v.
Ramverk	Limkod som får driver, runner och rapportering att fungera
Driver	Ett verktyg som interagerar med ett gränssnitt. T.ex. SoapUI, Selenium, Robot
Runner	Något som sköter förutsättningshantering för testfall, exekverar varje testfall och fångar resultaten av dem. T.ex. NUnit, JUnit, MS Test, Gallio
Byggmotor	Något som triggar kodbyggen och tester vid givna tidpunkter eller händelser. T.ex. Jenkins, TeamCity, MS Release Manager
Testfas	Tidsmässigt bundna testaktiviteter, t.ex. acceptanstest, installationsverifiering
Testnivå	Hur mycket av systemet som exekveras: Enhetstest, integrationstest, systemtest, systemintegrationstest

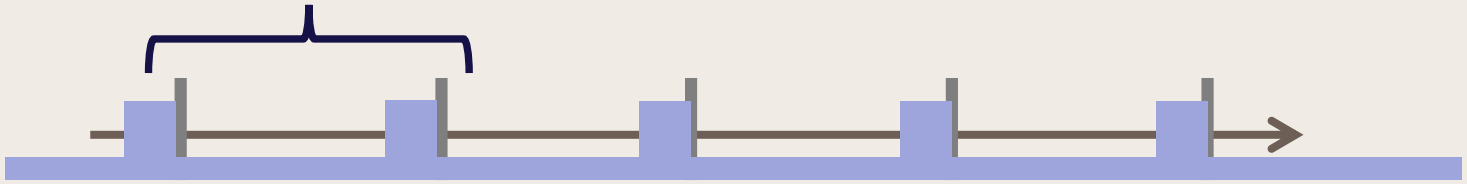
VARFÖR?

Control

- Visibility = ☺
- Steering = 😐

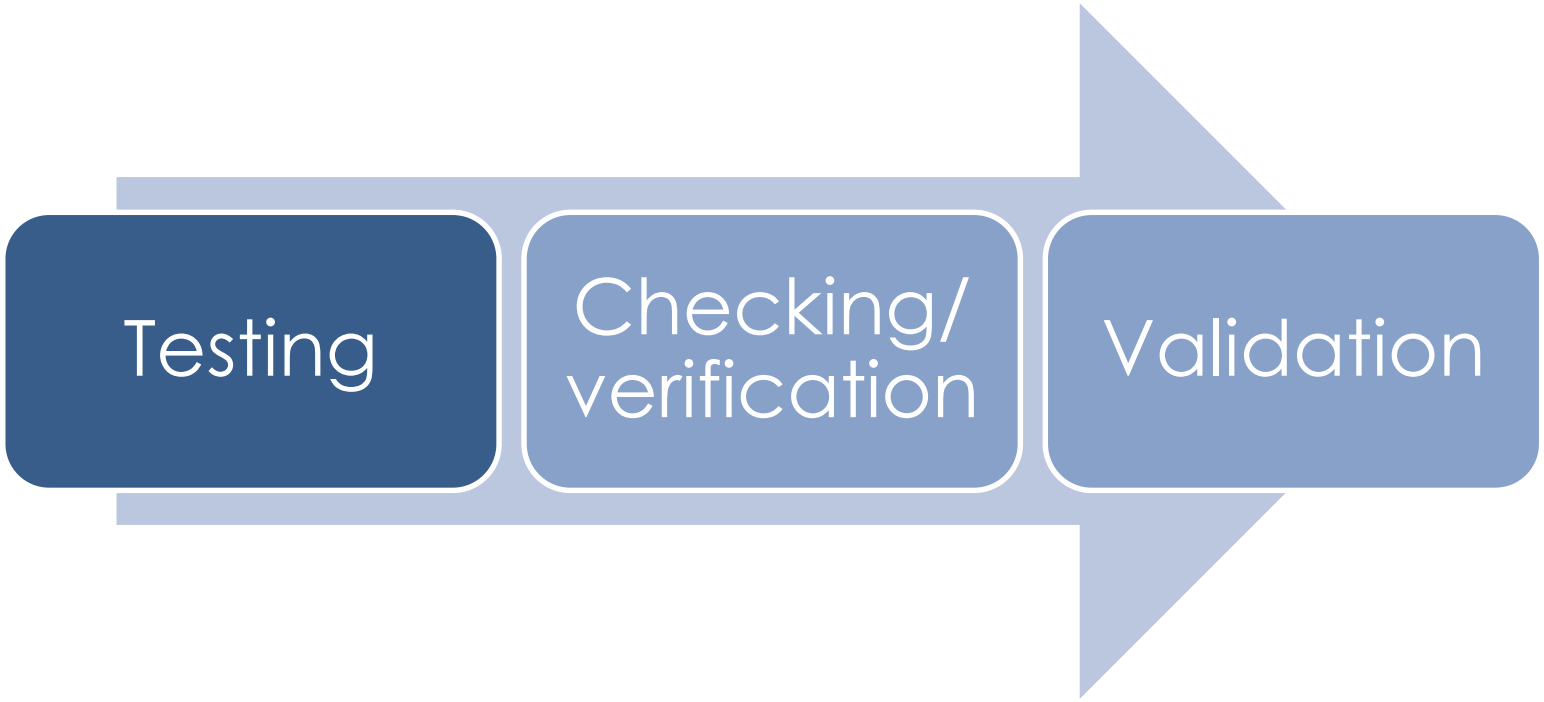
Hur fel kan det ha hunnit bli?

Minsta möjliga tid till värde

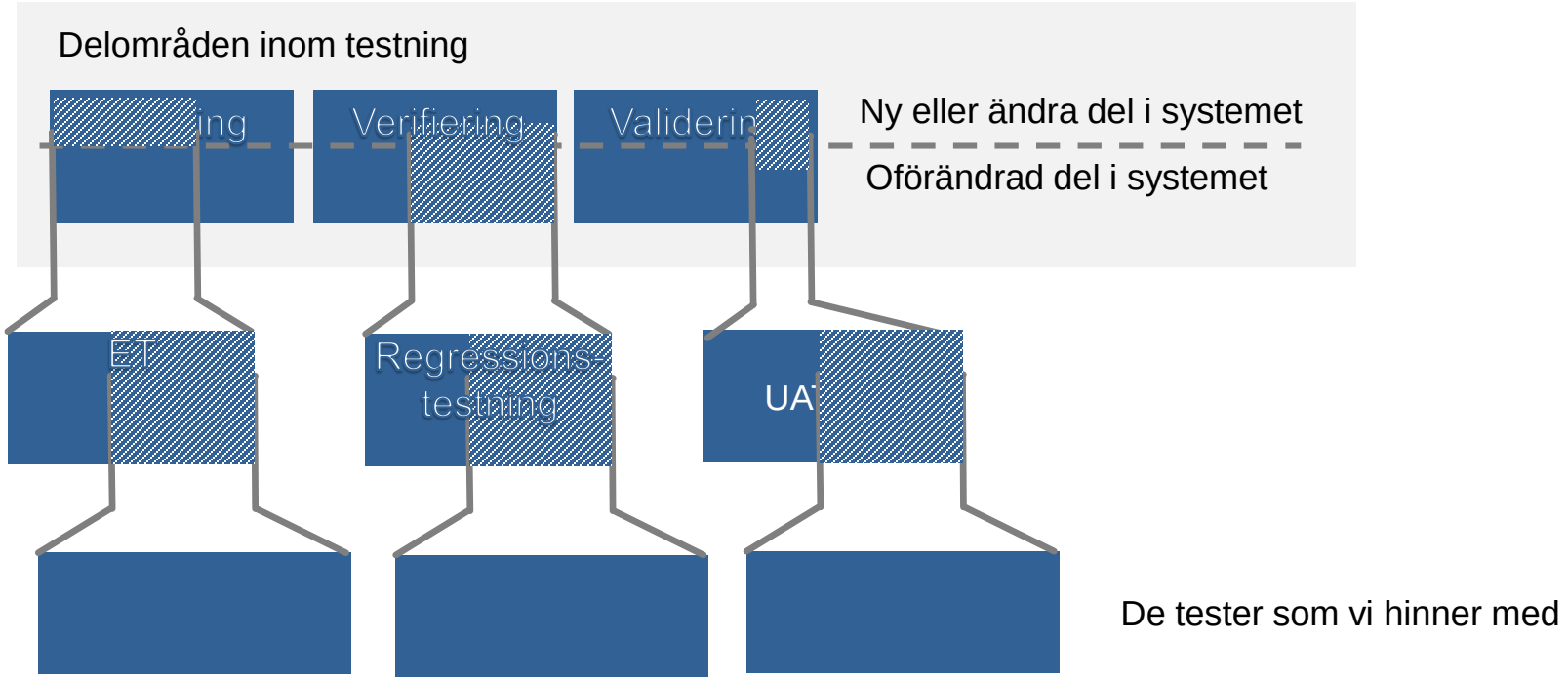


Minsta möjliga tid till värde

Hur fel kan det ha hunnit bli?

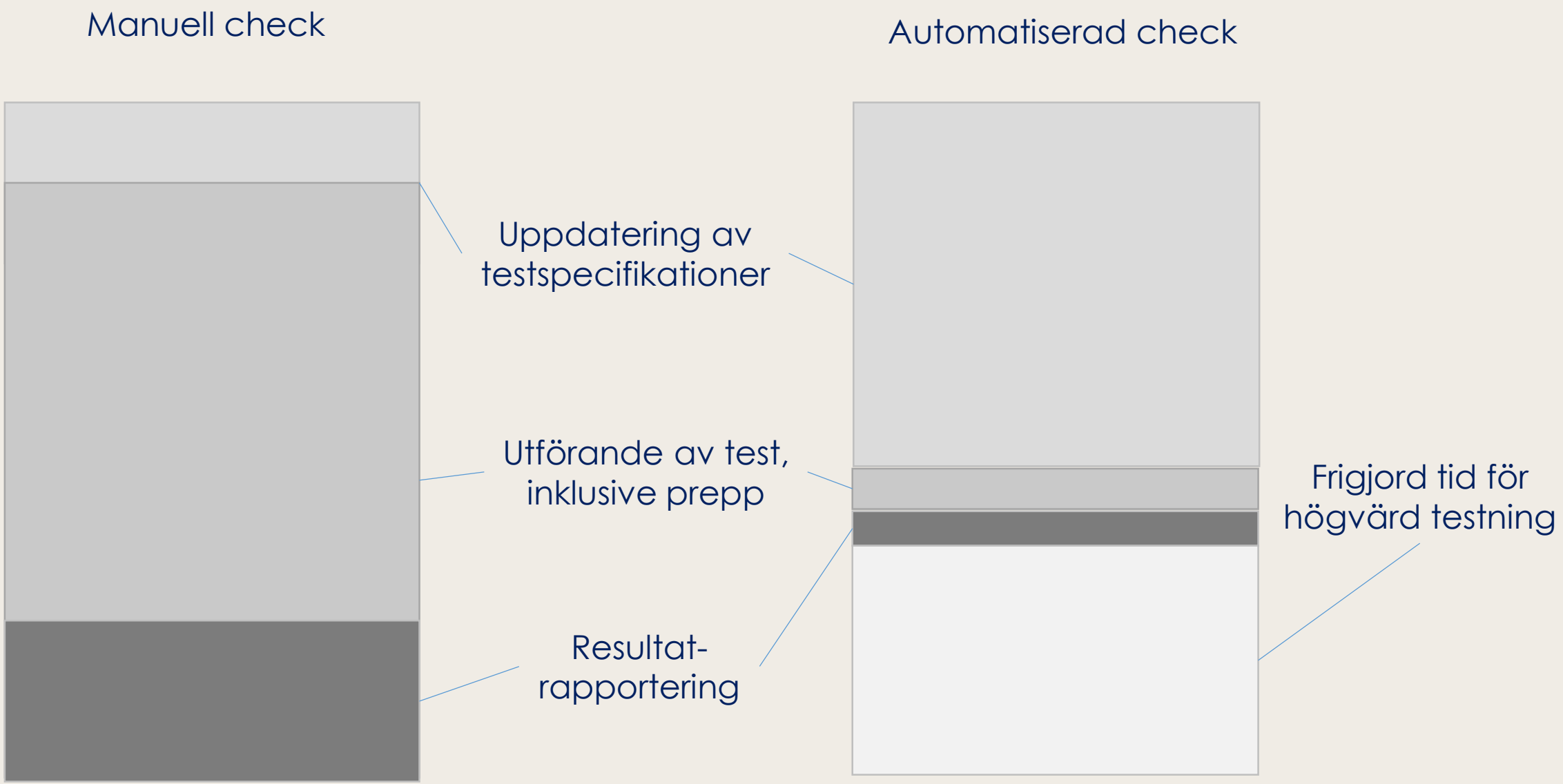

Testing
Checking/ verification
Validation

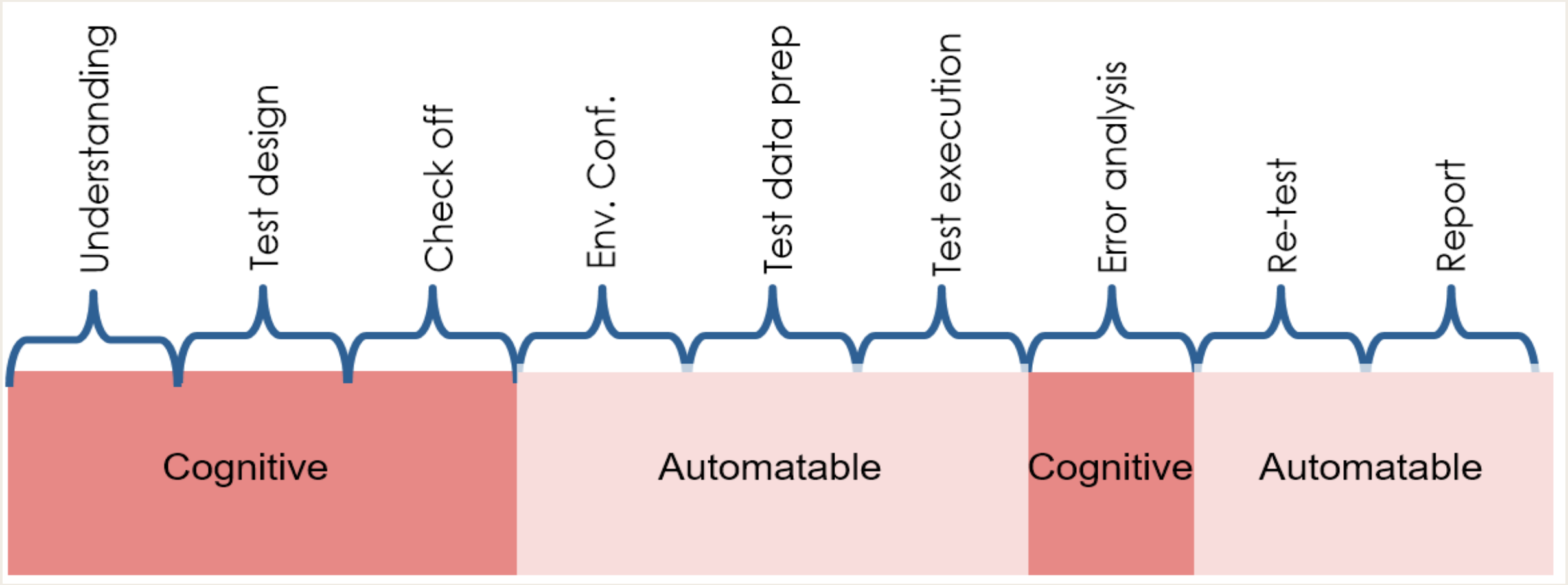
Testurval



De tester som vi hinner med

Manuellt arbete vid manuell checkning vs automatiserad checkning

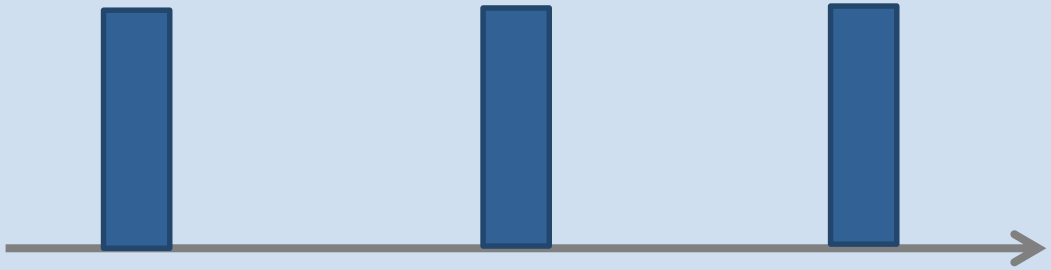



Understanding
Test design
Check off
Env. Conf.
Test data prep
Test execution
Error analysis
Re-test
Report
Cognitive
Automatable
Cognitive
Automatable

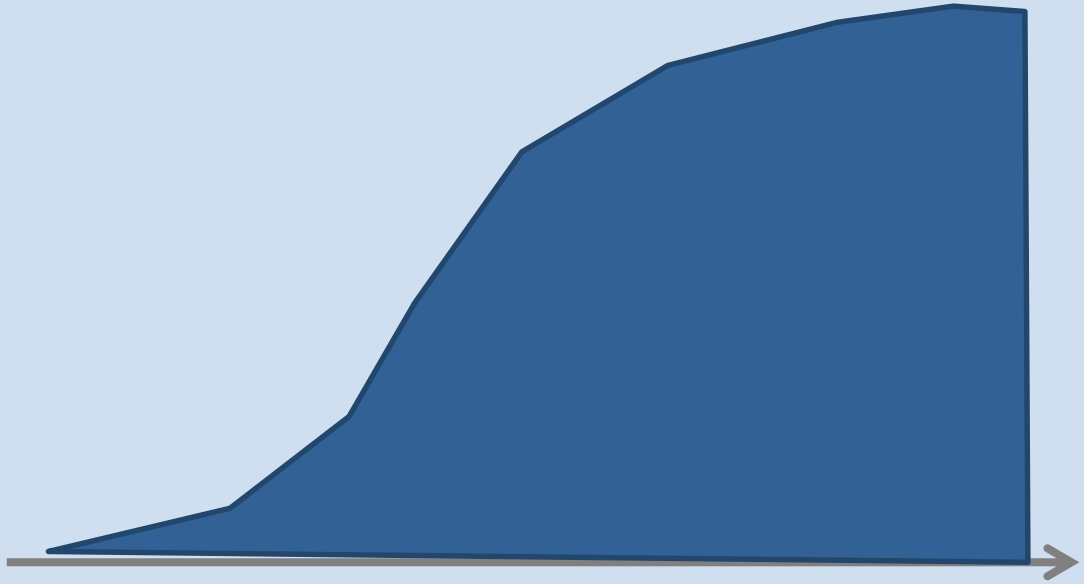
Manuell checkning vs automatiserad checkning

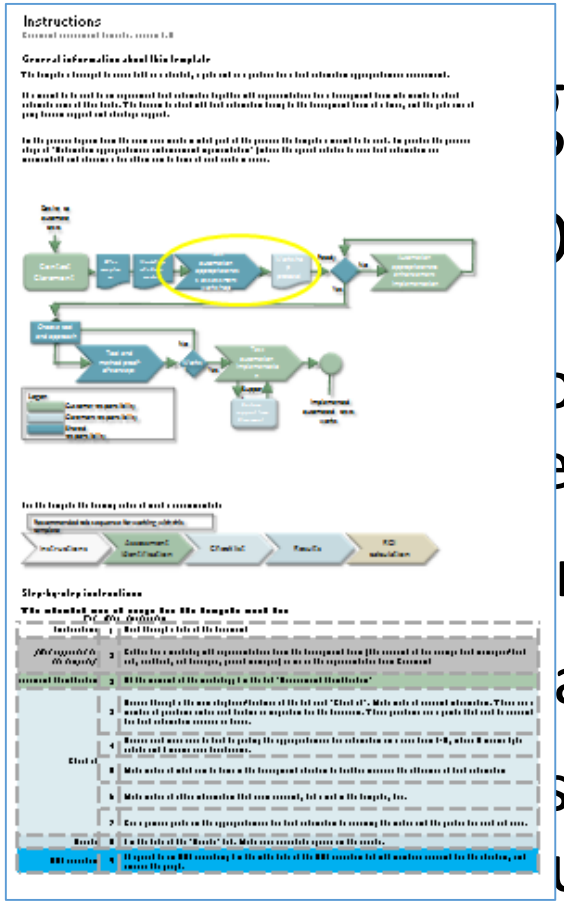
Antal utförda testfall över tid

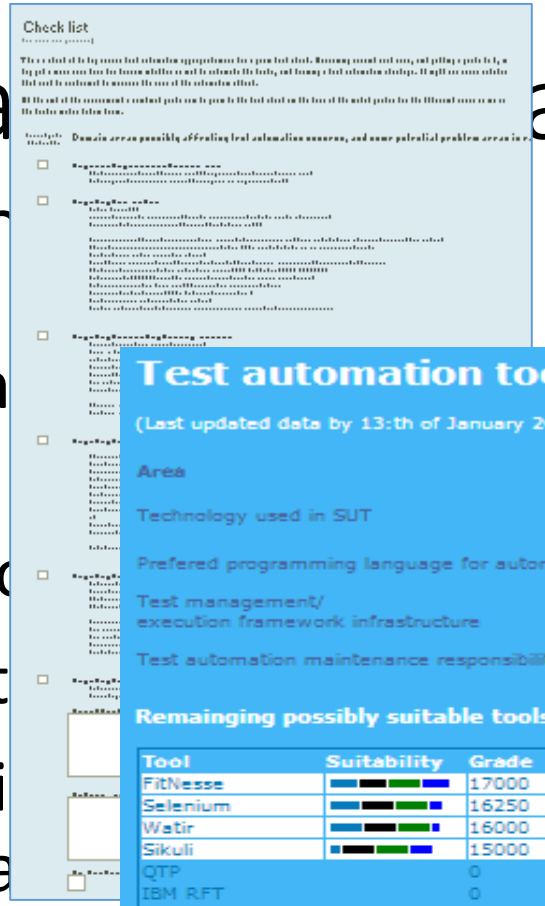
Manuell testning

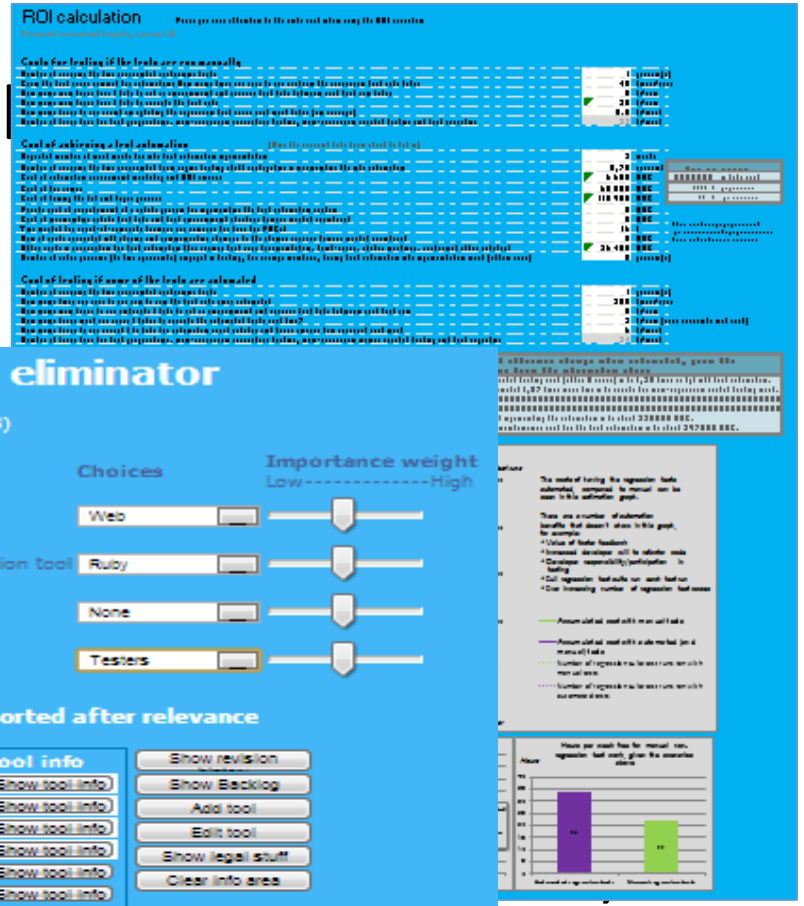


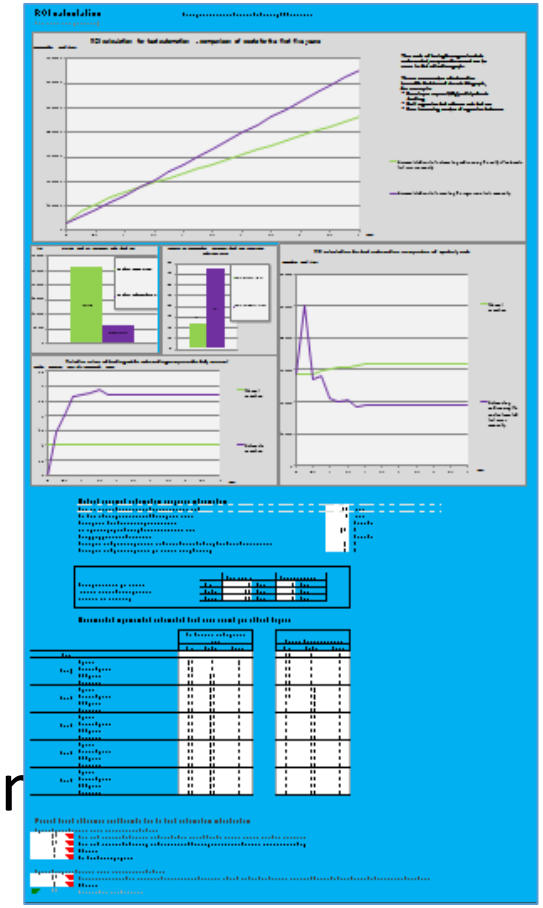
Automatiserad testning

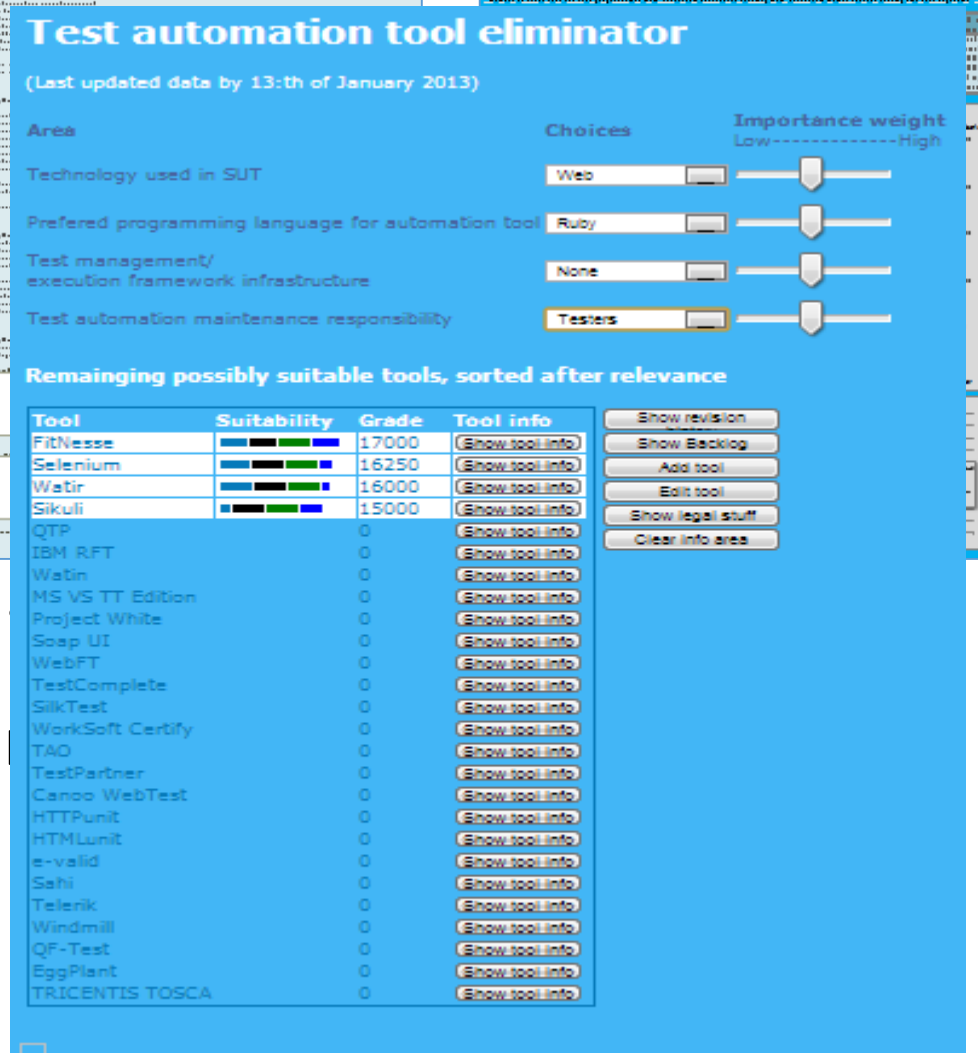












- Lämplig förvaltn
- Övriga förväntni

Frågor för testledare kring VARFÖR?

- Vad är syftet och effektförväntningen på testautomatiseringen? Är den rimlig?
- Finns det en roadmap för systemet under test som är så lång att automatiseringen kommer att vara givande under lång tid?
- Hur stor budget behövs? Smygs det in miljö- eller testdata-arbete i automationsimplementationen?
- Vilka intressenter finns det på testautomatiseringen? Detta styr vilken typ av information man vill ha ut av testautomatiseringen (vilka typer av rapporter o.s.v.).
- Hur ofta är testautomatiseringen tänkt att exekvera testerna? Är det typ var tredje månad är det sällan någonting värt (det kan ändå vara värdefullt om man måste kunna panik-patcha vrålsnabbt med bibehållen trygghet).

VAD?

Serendipitet

Regressionstester

Huvudflöden, kritisk funktionalitet

Enkla utforskande tester

Site-crawler, W3C check, kontroll av konfiguration

Testdataprepp

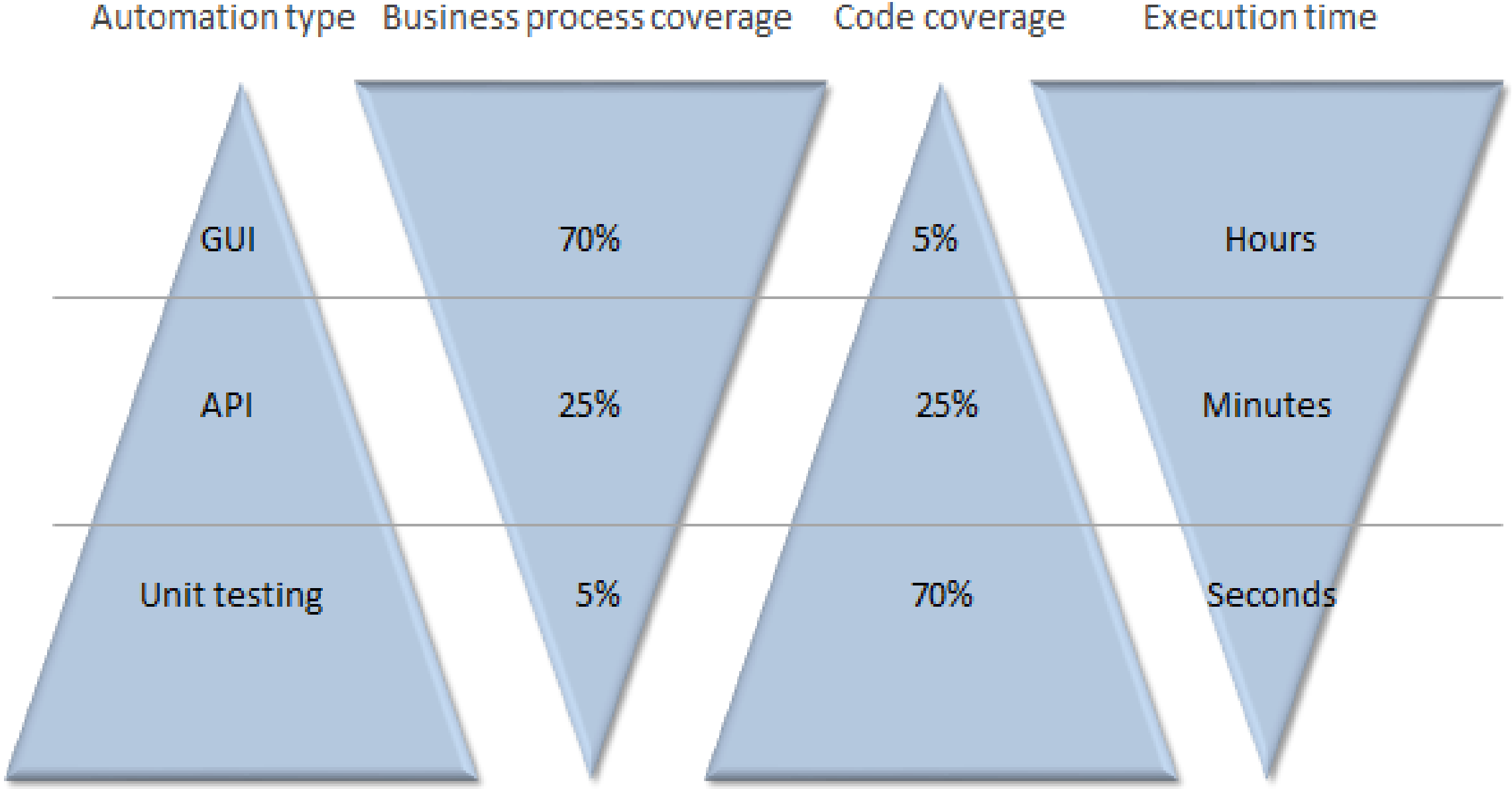
Miljö-deploy och -konfiguration

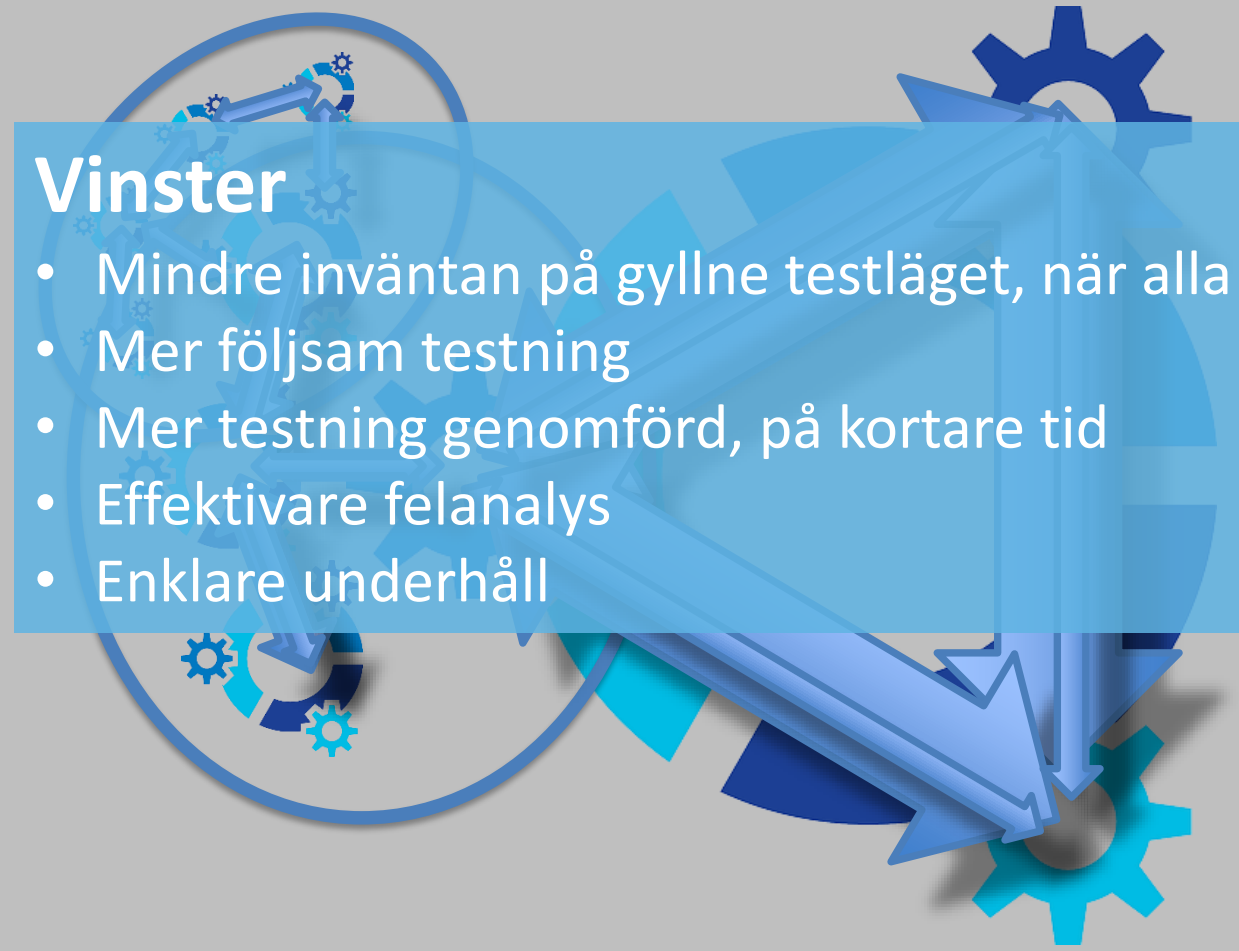
Problem: Närhets-bias

Den enda aktivitet som är värdeskapande inom test är när vi testar.

Allt annat bör automatiseras - om det går.

Vi vill inte ha några ”change notifiers”. Vi vill ha tester.


Automation type
Business process coverage
Code coverage
Execution time
GUI
70%
5%
Hours
API
25%
25%
Minutes
Unit testing
5%
70%
Seconds



Vinster

- Mindre inväntan på gyllne testläget, när alla system är i rätt status
- Mer följsam testning
- Mer testning genomförd, på kortare tid
- Effektivare felanalys
- Enklare underhåll

MANDAT

Undvik:

Undvik att tjata på utvecklare om enhetstester i väldigt gamla system. Det är förknippat med för stor risk och möda med att bygga om hela systemet så att det blir testbart med enhetstester. Detsamma gäller om systemet är byggt av alltför dåliga programmerare idag.

Undvik att skapa automatisering mot system som kan ändra sig hur som helst när som helst. Hela investeringen i automatisering kan förstöras i ett slag. Typexempel: Outsourcing.

Gör istället:

Föreslå att införa ett ramverk för Dependency Injection, om det går.

Föreslå att prova search-based-testing-verktyg som automatiskt skapar enhetstester.

I avtalen: Kravställ på testtäckningsgrad och hur kort tid det ska kunna ta för att få tillhanda en minimal patch-release med full testtäckning på hela systemet.

Använd GUI-verktyg som struntar i teknologin.

Kösta’ re’ nå’e?

Vad kostar en mantimme?

- Vad kostar felsökningstimmar för att leta orsaker till buggar i gammal kod?
- Vad kostar förseningar för att man har långa testperioder och omtestperioder?
- Vad kostar personalersättning för att teammedlemmar har tråkigt på jobbet?
- Vad kostar inflexibiliteten att inte snabbt kunna få ändringar till produktion på ett tryggt sätt?
- Vad kostar det att utvecklare och testare inte kommunicerar bra?

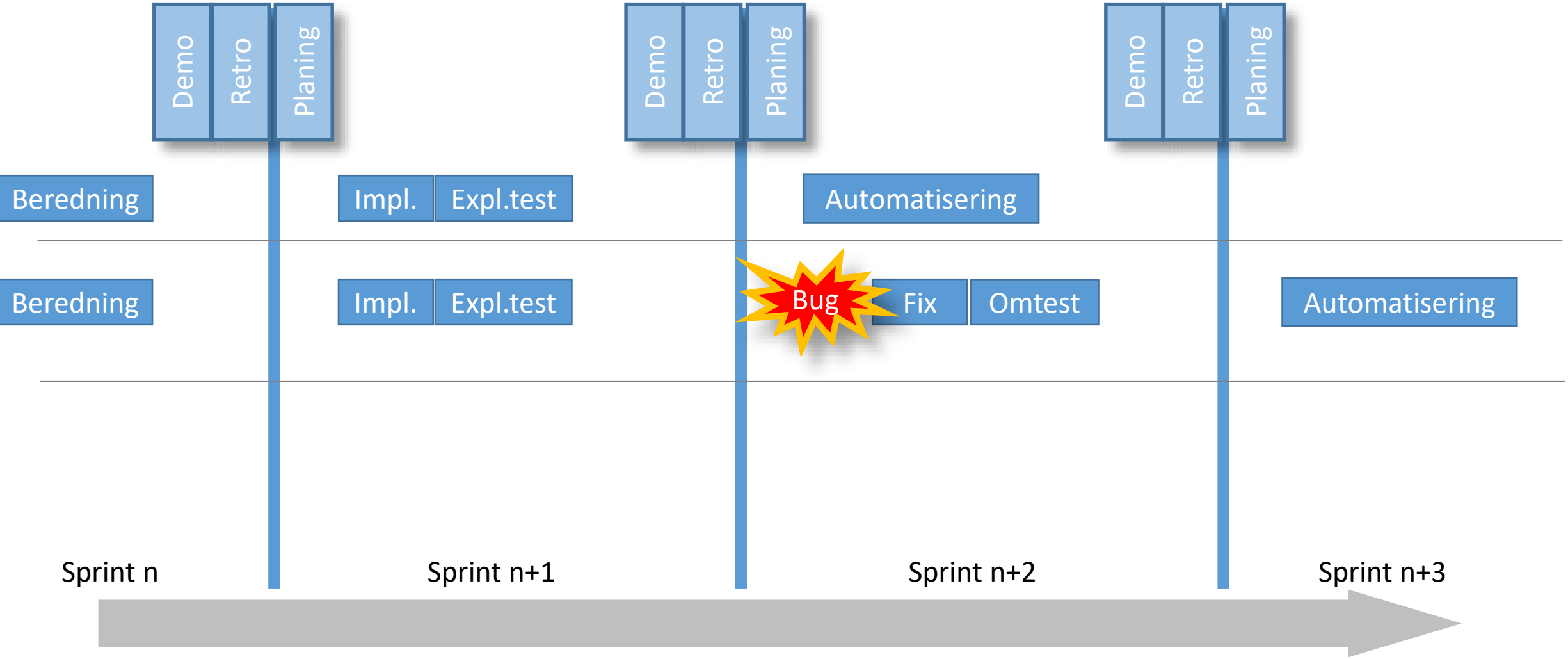
Bra frågor för testledare kring VAD?

- Vilka typer av tester lämpar sig att automatiseras i detta projekt?
- Vilka kompetenser finns det att tillgå, hur mycket tid kan de undvara? Det kan styra vilka tester man väljer att automatisera.
- Finns det tidigare testautomatiseringsinitiativ som misslyckats? Varför misslyckades dessa?
- Vilka testfall vill testarna slippa utföra? Vilka testfall är viktigast?

NÄR?

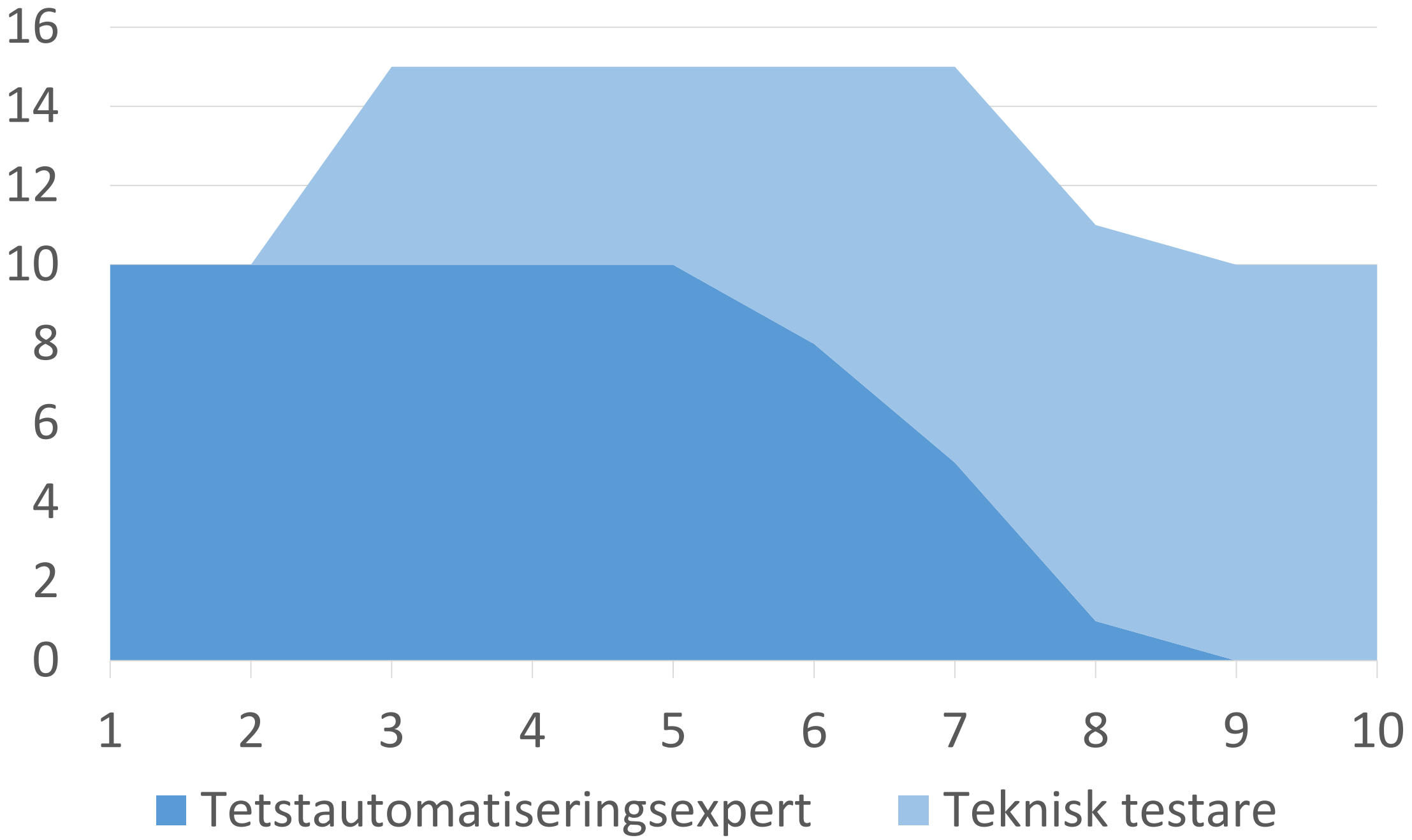
DevOps + Agilitet

Graf över när i tiden…



Te[illegible]veckling (bland utvecklare)

1. F[illegible]ktionalit[illegible]en först.
2. Utveckla testfa[illegible]en färdiga [illegible]en
3. Provkör testfa[illegible]e fallerar
4. Utveckla tills tes[illegible]ger[illegible] och komplettera testfallen i takt med att man inser att funktionaliteten behöver berikas.


16
14
12
10
8
6
4
2
0
1
2
3
4
5
6
7
8
9
10
Tetstautomatiseringsexpert
Teknisk testare

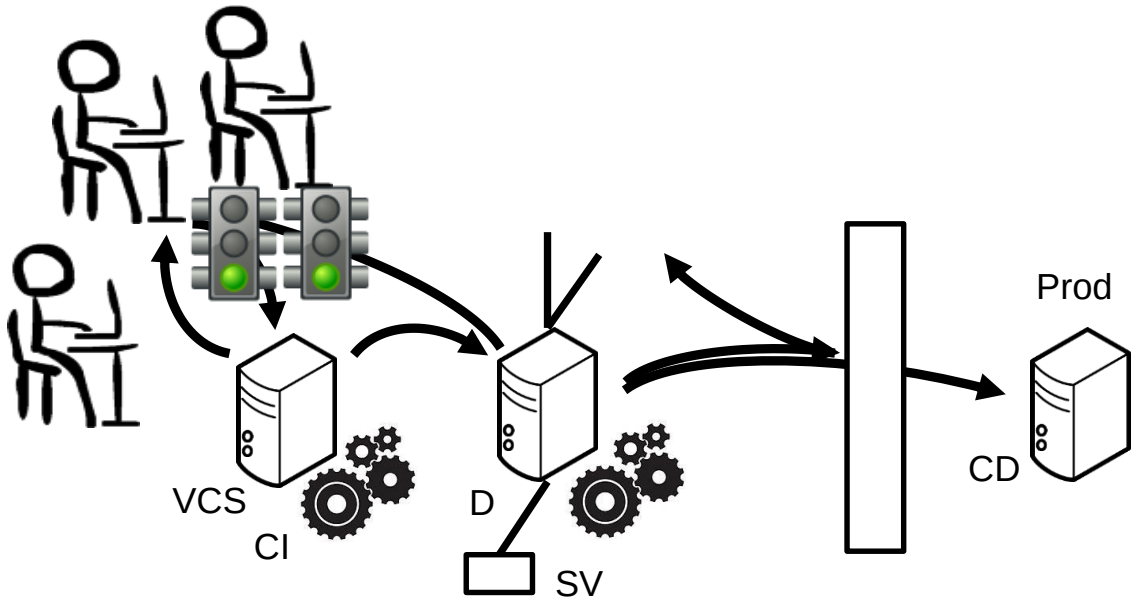
Bra frågor för testledare kring NÄR?

- Vilka testfall finns i en status så att de är redo att automatiseras mot? Vilken testnivå är dessa på?
- Finns resurser tillgängliga som kan implementera?
- Finns resurser tillgängliga som kan svara på frågor?
- Är funktionaliteten någorlunda stabil i sina huvuddrag så att scriptunderhållet blir hanterbart med tanke på förändringstakten?
- Finns det lämpliga testmiljöer att exekvera testerna i?
- Har man en strategi framtagen för att hantera testdata på ett sätt som fungerar för testautomatiseringen?
- Vill man koppla testautomatiseringens resultat till ett test-management-system, och finns då testfall och krav däri?

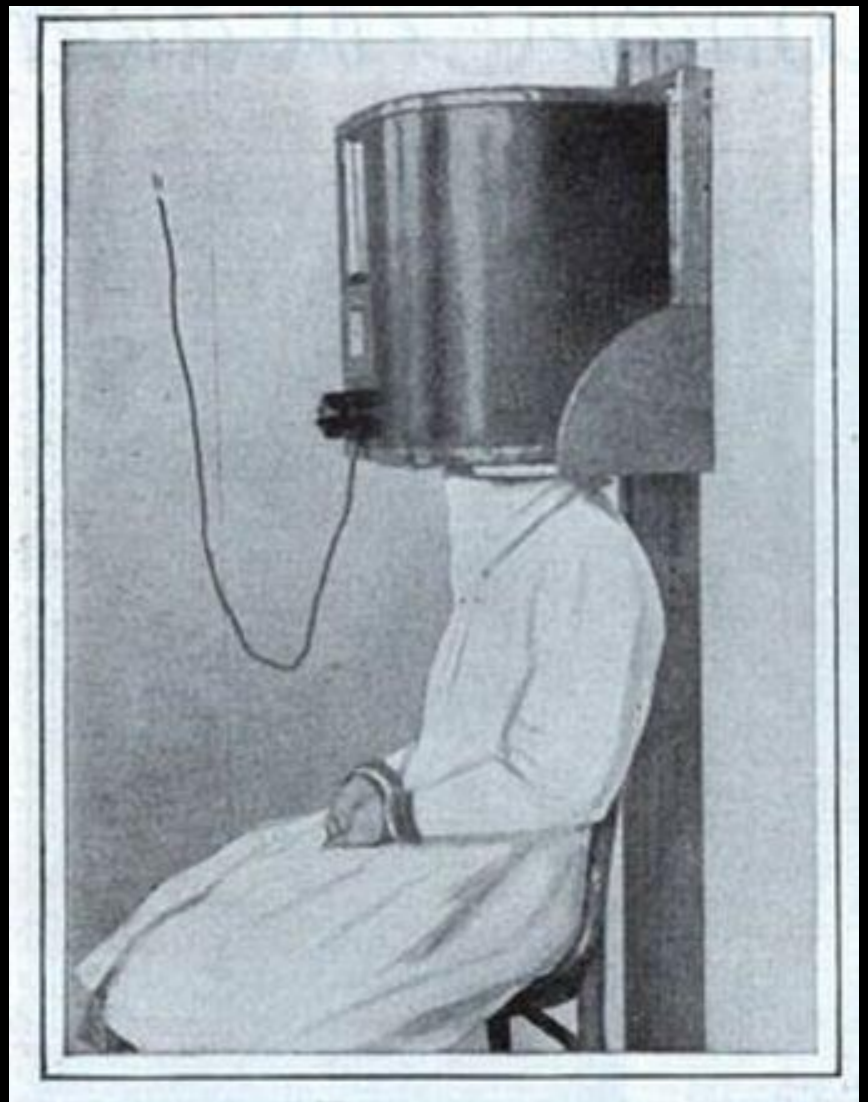
VEM?

Continuous Delivery

Continuous Integration
Continuous Deploy



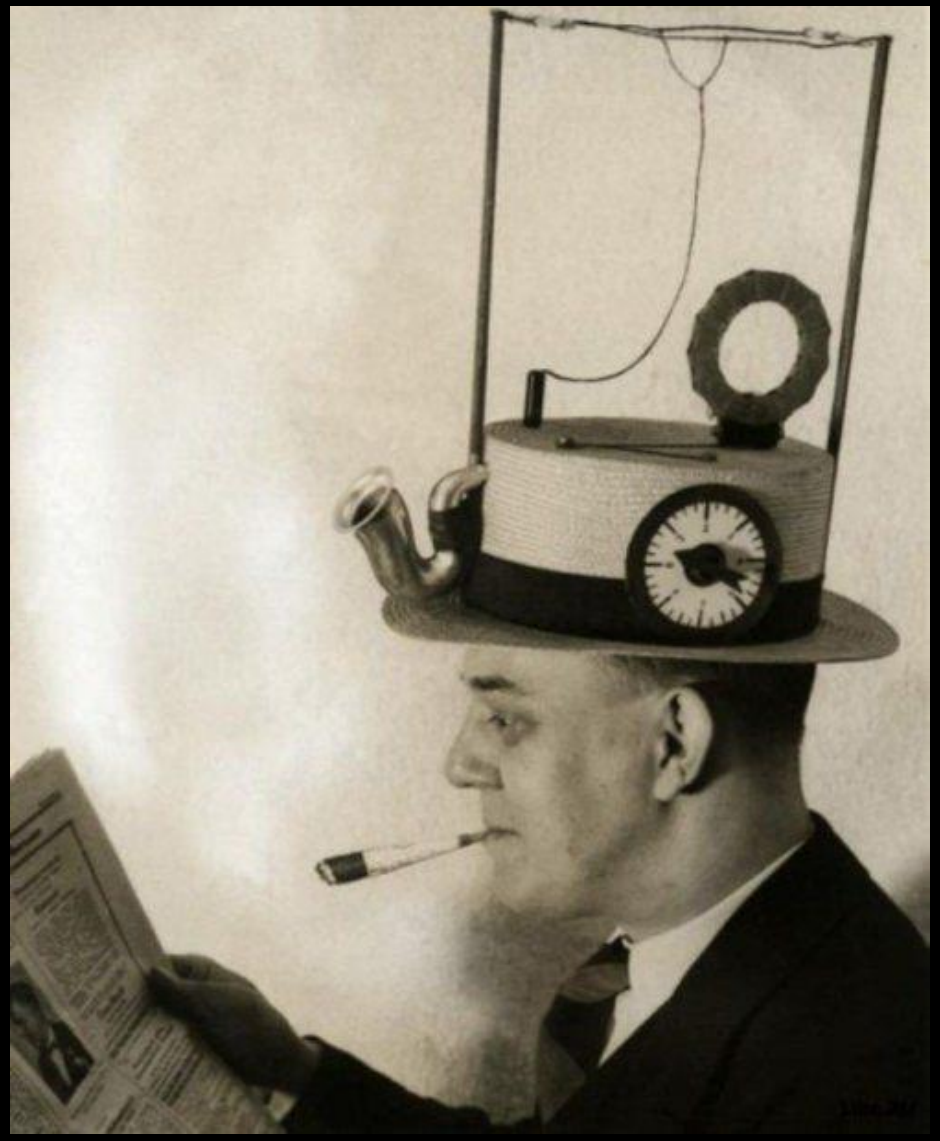
Kodande utvecklare



- Superkraft i att skriva underhållbar kod!
- Testning blir nerprioriterat
  - ”Vi kan bygga ett eget ramverk”
  - …men tar sig inte för att göra det…
- Dålig testning
  - Genvägar för att få snygg kod
  - Täcker bara det de redan har täckt
  - Asserts i systemtest…
- Ovana vid att hantera alla utmaningar som testare möts av
  - De ger upp och tycker att automatisering är jobbigt

Den tekniske testaren

- Förstår ungefär vad kod gör när den läser den
  - …men är inte superbekväm i att skriva kod
- Vill egentligen ägna tiden åt högvärd testning
- Har mycket att vinna på testautomation på systemtestnivå – kommer att se till att underhållet görs



- En person som ska vara sin egen arkitekt för en kodbas med mer beroenden till andra system än något annat
- En person som ska kunna kommunicera med både verksamhet, testare, arkitekter och utvecklare
- En person som ska kunna vara självstyrande och planera sitt eget arbete
- En person som ska ha förståelse för både testning och utvecklingsprocessen

Bra frågor för testledare kring VEM?

- Hur ser den tänkta förvaltningsorganisationen för testautomatiseringen ut? Det styr hur den implementeras?
- Vem besitter både testkompetens, verksamhetsförståelse och utvecklingskompetens? Går det att kompromissa med någon av dessa med stödåtgärder av andra människor?
- Hur ser samarbetet med utvecklarna ut? Vad tycker de om test? Hur ser de på automatisering?

HUR?

Testautomatiseringutmaningar

- Ändringar i gränssnitt kan ge avvikelser
- Ändringar i processer kan ge avvikelser
- Ändringar i teknologi kan förstöra helt
- Timing-problem kan ge avvikelser
- Defekter i SUT kan ge avvikelser

Det tar längre tid att underhålla och analysera resultatet än vad som egentligen känns givande

Organisatoriskt svårplacerat

Testautomatisering är tidsödande

- Analysera avvikelsers orsak tar tid
- Ändra testfall och applikationsbeskrivningar tar tid
- Implementera ramverk tar tid

Det största hotet mot testautomatisering är att det är så kul

Implementationsstrategi

1. Förstå testernas kringmiljö och förstå testfall nummer 1 (ganska avancerat testfall så att det skapar grunden för en god arkitektur från början)
2. Skapa ett anpassat ramverk medan man implementerar första testfallet
3. Anpassa ramverket vidare medan man implementerar fler testfall – i ordning baserade på testområde
4. Kontinuerligt dokumentera lösningen för att kunna möjliggöra att anpassa antalet involverade personer
5. Kontinuerligt mäta progress och penetration – och kommunicera detta till berörda

GUI test automation killers

Brittle:

- Changes to GUI
- Changes in workflow
- Changes in technologies

Time consuming:

- Overwhelming maintenance
- Time consuming to analyze test results

Confidence in test results

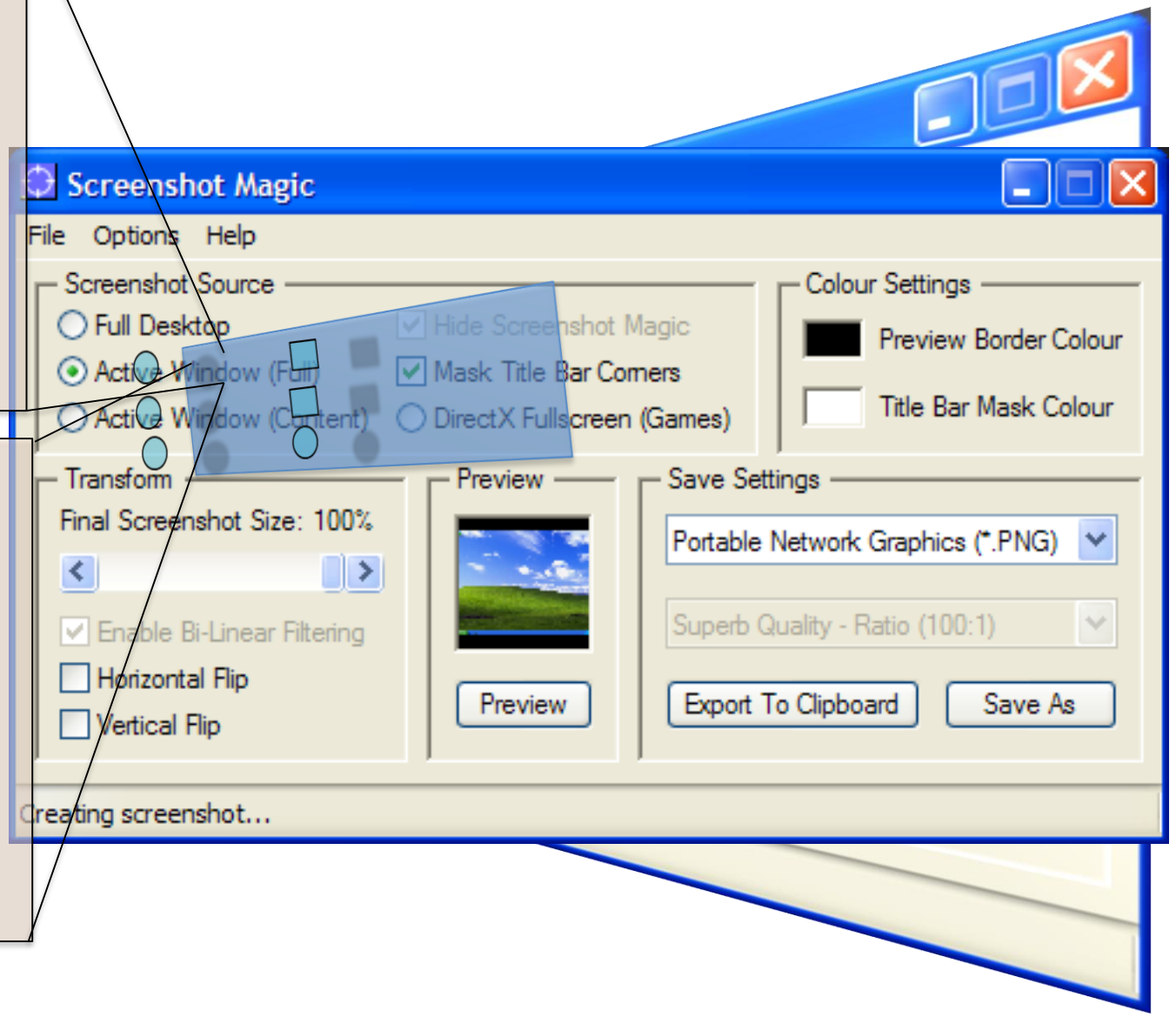
- False errors

Organization:

- No responsibility
- Unfitting

Properties/
Egenskaper

Methods/
Metoder



”Moderna”

Model Based Testing
Keyword driven test a
Datadrivna tester
Specification By Exam

```
Scenario: Free shipping in Australia
Given I am on the Beautiful Tea home page
When I search for ‘Byron Breakfast’ tea
Then I see the page for ‘Byron Breakfast’ tea
When I add ‘Byron Breakfast’ tea to my cart
And I select 10 as the quantity
Then I see 10 x ‘Byron Breakfast’ tea in my cart
When I select ‘Check Out’
And I enter my country as ‘Australia’
Then I see the total including GST
And I see that I am eligible for free shipping
```

	A	B	C
1	Name	Product	Quant
2	John Smith Jr	MyMoney	
3	Clare Jefferson	MyMoney	
4	Susan McLaren	MyMoney	50
5	Charles Dodgeson	FamilyAlbum	1
6	Steve Johns	FamilyAlbum	10
7	Samuel Clemens	FamilyAlbum	50
8	Bob Feather	ScreenSaver	1
9	Mark Smith	ScreenSaver	10

5/12/2000	7, Flower Street	Earlcastle	
5/12/2000	45, Stone st.	Bringtone, TX	
12/14/2003	17, Park Avenue	Salmon Island	
12/12/1999	3, Garden st.	Hillsberry, UT	
12/12/2005	14, North av.	Milltown, WI	
10/10/2006	9, Maple Valley	Whitestone, British Columbia	

Sikuli X-1.0rc3 (r905) - Citrix_Speed_Test_v3.sikuli

File Edit Run View Tools Help

Take screenshot | Insert image | Create Region | Run | Run in slow motion | Find

Find

exists( )
find( )
findAll( )
wait( )
waitVanish( )

Mouse Actions

click( )
doubleClick( )
rightClick( )
hover( )
dragDrop(,)

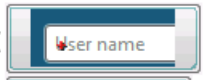
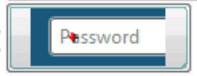
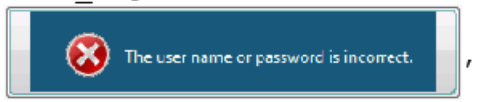
Keyboard Actions

type(text)
type(, text)
paste(text)
paste(, text)

Event Observation

onAppear(, handler)
onVanish(, handler)
onChange(handler)

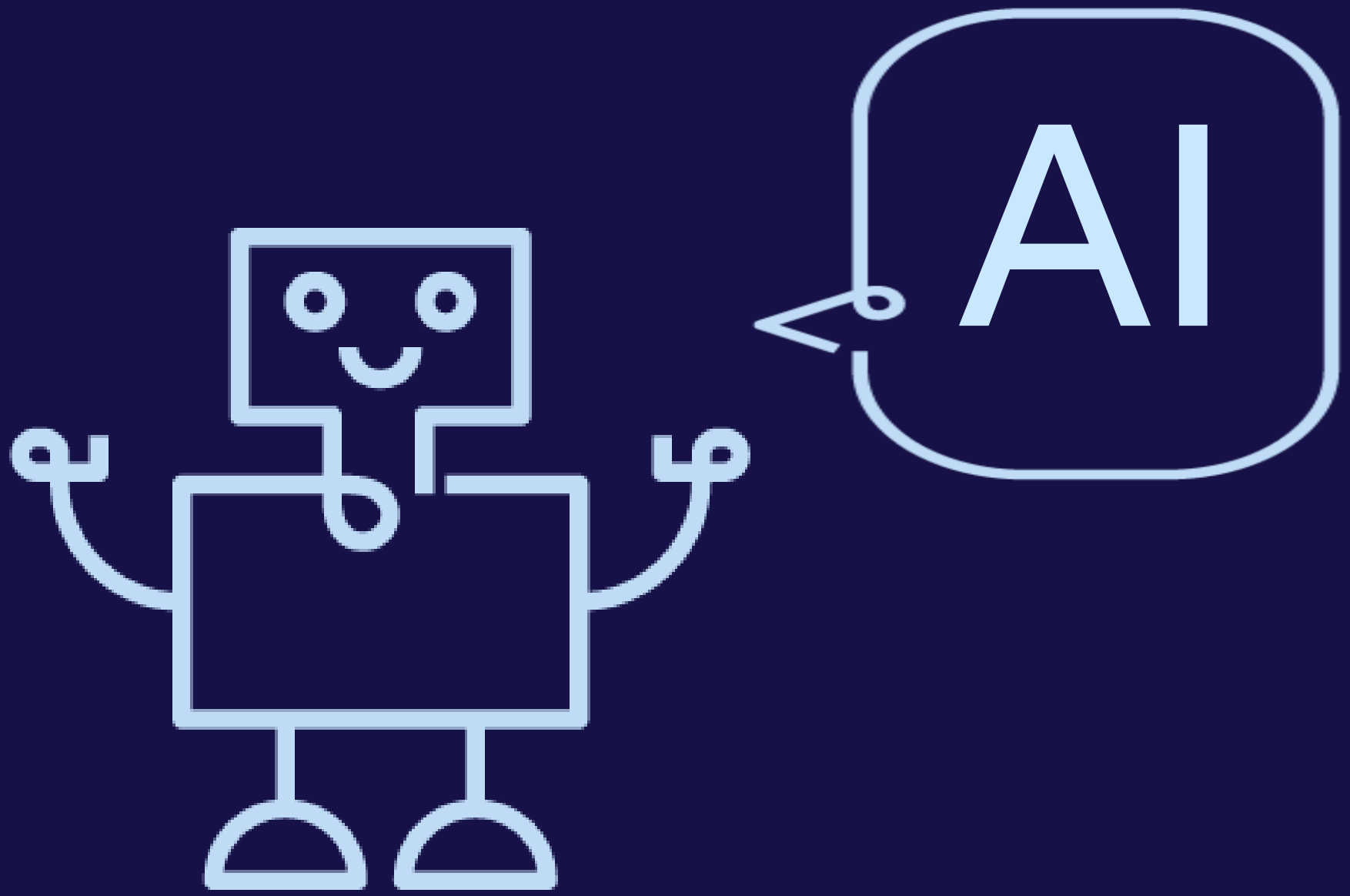
Citrix_Speed_Test_v3.sikuli | Untitled

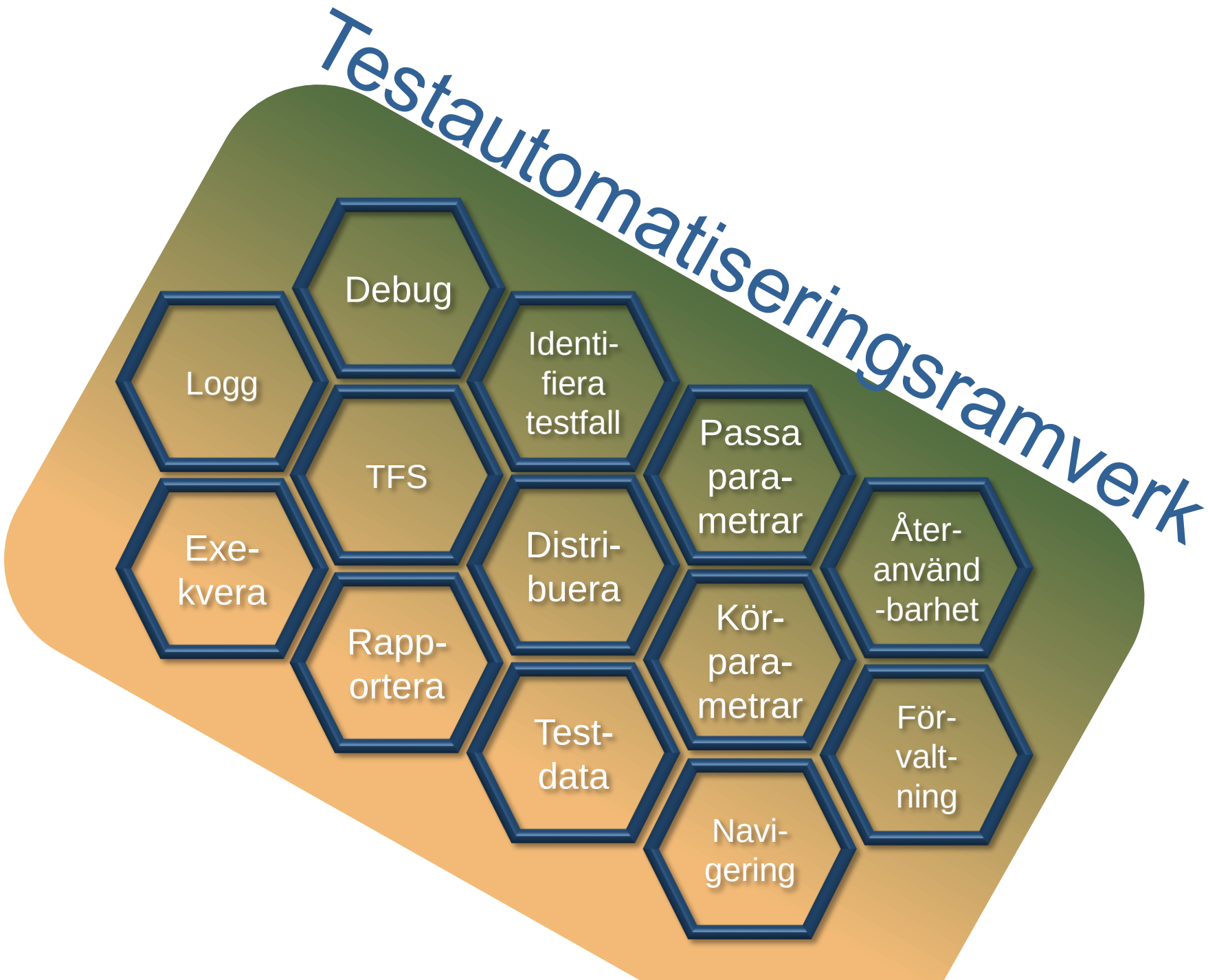
```
18
19 Xuser = input("Username to use for test")
20 Xpassword = input("Password for user " + Xuser)
21
22 faulty_login = 0
23 while not exists("",0):
24     t_start = time.time()
25     myApp = App.open(citrixExecutable) # Note, that the ica fi
26     wait(  ,60)
27     wait(1)
28     type(  , Xuser)
29     type(  , Xpassword)
30
31     click(  )
32     t_start_after_login = time.time()
33     onAppear(  , reset_after_wrong_pass
34     observe(25,False)
35
```

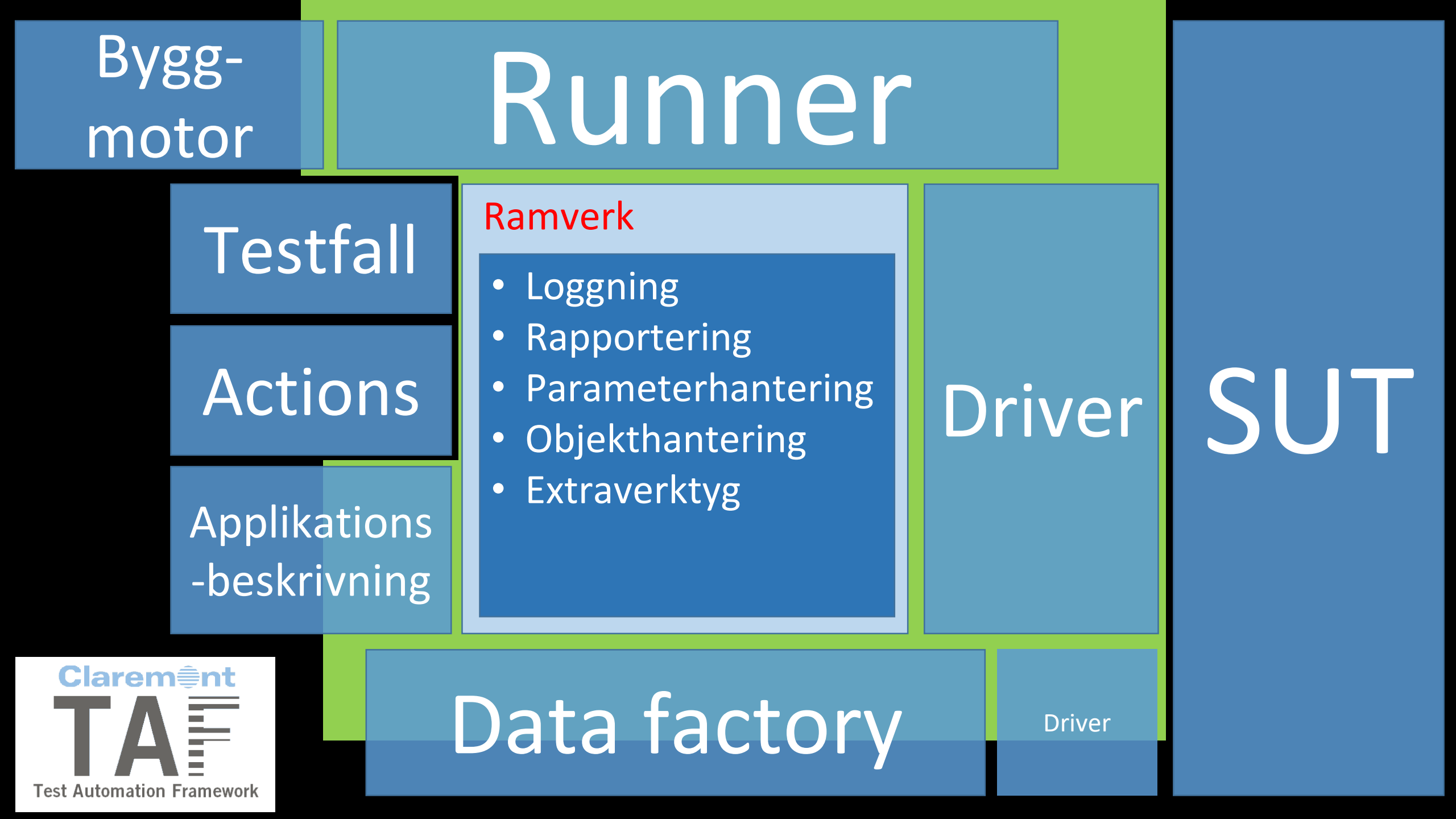

Message | Test Trace

```
[log] CLICK on (739,506)
[log] TYPE "olivere"
[log] CLICK on (743,536)
```

Line: 5, Column: 1


AI


Testautomatiseringsramverk
Debug
Logg
Identi-
fiera
testfall
TFS
Passa
para-
metrar
Exe-
kvera
Distri-
buera
Åter-
använd
-barhet
Rapp-
ortera
Kör-
para-
metrar
Test-
data
För-
valt-
ning
Navi-
gering


Bygg-
motor
Runner
Testfall
Actions
Applikations
-beskrivning
Ramverk
• Loggning
• Rapportering
• Parameterhantering
• Objekthantering
• Extraverktyg
Driver
SUT
Data factory
Driver
Claremont
TAF
Test Automation Framework

Statistics

YIPPIE!

Result	Count
Passed test cases	79

Bra frågor för testledare kring HUR?

- Vem ska förvalta testautomatiseringslösningen?
- Finns det ett specifikt verktyg för automation i organisationen (prefered supplier)
- Finns det ett ramverk framtaget som används någonstans.
- Hur är viljan till automatisering och synen på test bland utvecklarna?

Mätetal?

Mätetal kring testautomatisering

- Antal personer per vecka som commitat kod till testautomatiseringen
- Antal testfall som aldrig fallerat
- Antal testfall som fallerar ofta
- Antal implementerade testfall
- Antal timmar lagda på scriptunderhåll

"The numbers have no way of speaking for themselves. We speak for them. We imbue them with meaning."

–Nate Silver, The Signal and the Noise

PageObject Model

Fallgropar

Införande

Alternativ

Pairwice testing

MBT

ATDD

UT

Bygghantering

TDD

BDD

Deployprocess

Framgångsfaktorer

SIT

Specification-by-example

ST

Historik

Datahantering

Utmaningar

Tips och trix

Miljöer

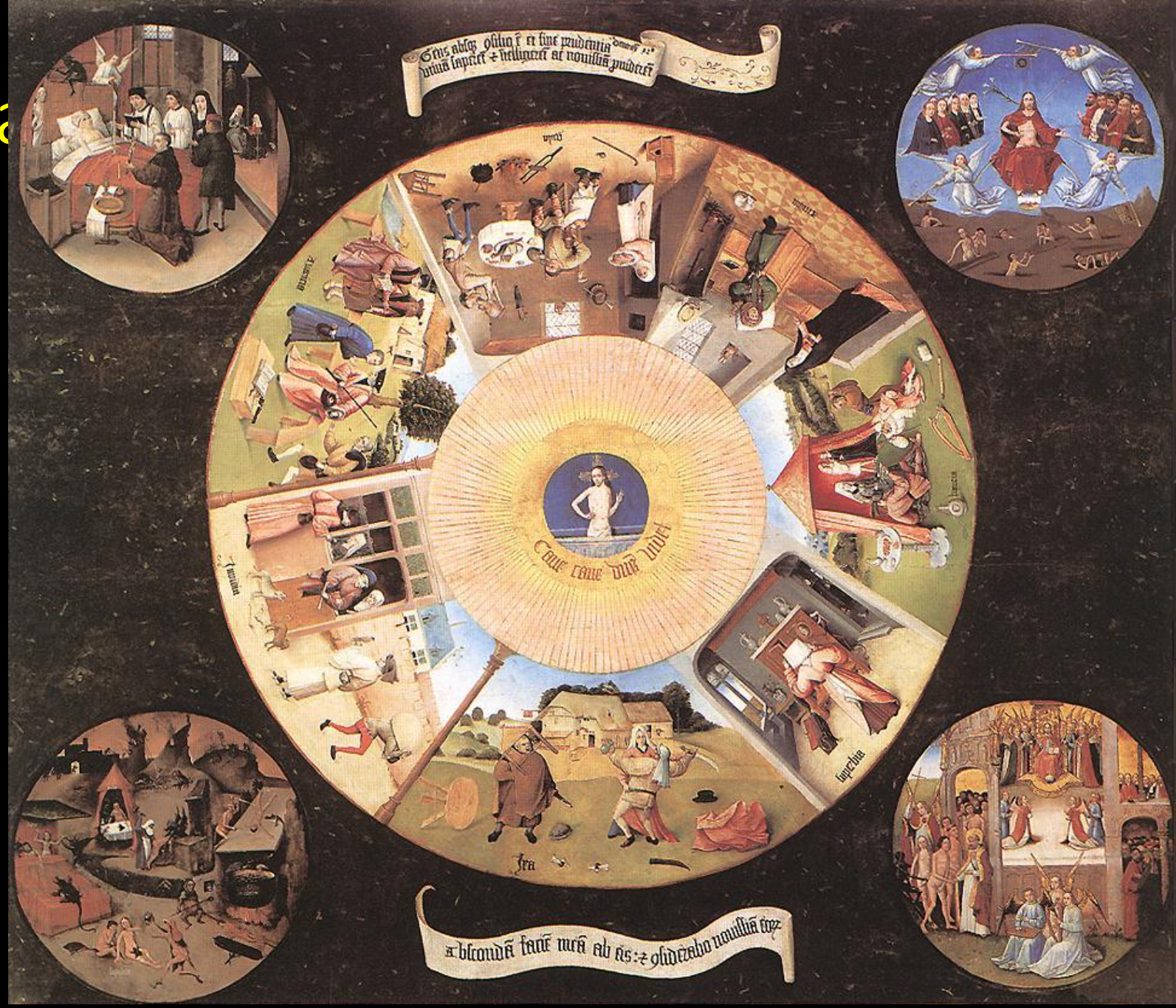
Verktygsval

Etc

Lämplighet och ROI

De sju dödssynderna

1. Pride/Högmod/Superbia
2. Lust/Lust/Luxuria
3. Envy/Avund/Invidia
4. Gluttont/Frosseri/Gula
5. Wrath/Vrede/Ira
6. Greed/Girighet/Avaritia
7. Sloth/Lättja/Acedia



Hieronymus Bosch's The Seven Deadly Sins and the Four Last Things

Kod är en kostnad.

Leverera bevisad funktionalitet.

Q 

Frågor?

Tack!

Tack till:
Er, såklart
Zington (för att de lät mig få ledigt för detta)
David Pers, som är hyvens och låter mig använda Tore Testledare
Mina kollegor genom åren för alla lärdomar
Memebase (för sköna bilder)
Pictureisunrelated.com (för fler bilder)

jorgen.damberg@zingtongroup.com

J:Damberg

cognitivedistortion.com