
TAF
Test Automation Framework

Q

Software quality characteristics

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 något den inte ska).
- Interoperabilitet: olika funktioner interagerar med varandra på bästa sätt.
- Samtidighet: möjlighet att utföra multipla parallella uppgifter, och att kunna köra samtidigt som andra processer.
- Dataagnostisk: stödjer allehanda dataformat och hanterar databrus.
- 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.
- Dataintegritet: all sorts data behålls intakt genom hela produkten.
- Säkerhet: produkten medverkar inte till att skada personer eller egendom.
- Katastrofhantering: vad händer om någonting riktigt, riktigt allvarligt inträffar?
- Trovärdighet: är 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.
- Ihågkommbarhet: 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 användargränssnittet.
- Snabbanvänd: en erfaren användare kan utföra vanliga uppgifter väldigt snabbt.
- Interaktivitet: produkten har lättförståeliga tillstånd och utmärkta möjligheter till interaktion via användargränssnittet.
- Kontroll: användarens känsla av kontroll över programvaran genom hela användarscenariot.
- Klarhet: är allting explicit och detaljerat uttryckt, med ett språk som kan bli förstått och inte är tvivelaktigt?
- Felhantering: felmeddelanden är informativa, det är svårt att göra misstag och lätt 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?
- Över förväntan: produkten överträffar dina förväntningar och möter de behov du inte visste du hade.
- Attityd: har produkten och dess information den rätta attityden och pratar till dig med rätt språk och stil?
- Förstaintryck: är (första) intrycken av produkten imponerande?
- Historier: finns det intressanta historier om produktens uppkomst, konstruktion eller användande?

Säkerhet. Stoppar produkten oönskat användande?
- 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 systemen.
- Virusfri: produkten ska inte vidarebefordra virus eller ge intrycket av att vara det.
- Piratkopieringsskydd: ingen möjlighet att illegalt kopiera och distribuera programvaran eller koden.
- Reglementsenlig: produkten uppfyller tillämpliga säkerhetsstandarder.

Prestanda. Är produkten tillräckligt snabb?
- Kapacitet: produktens möjligheter/begränsningar i många olika situationer (t.ex. slött nätverk)?
- Resursnyttjande: 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 köra på stödda konfigurationer och att hantera olika miljöer eller saknade komponenter.
- Installerbarhet: produkten kan installeras på avsedda plattformar och använder en rimlig storlek av hårddiskutrymmet.
- Uppgradering: det är enkelt att uppgradera till en ny version utan att tappa konfigureringsdata och inställningar.
- Avinstallation: alla filer (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 omgivningens konfigurationer.
- Bakåtkompatibilitet: kan produkten göra allt den förra versionen kunde?
- Framåtkompatibilitet: kommer produkten att kunna använda artefakter eller gränssnitt i framtida versioner?
- Hållbarhet: påverkan på miljön, t.ex. energieffektivitet, elbesparingslägen, möjlighet till distansarbete.
- Reglementsenlig: produkten uppfyller applicerbara standarder, regelverk, lagar, riktlinjer eller etiska förhållningssätt.

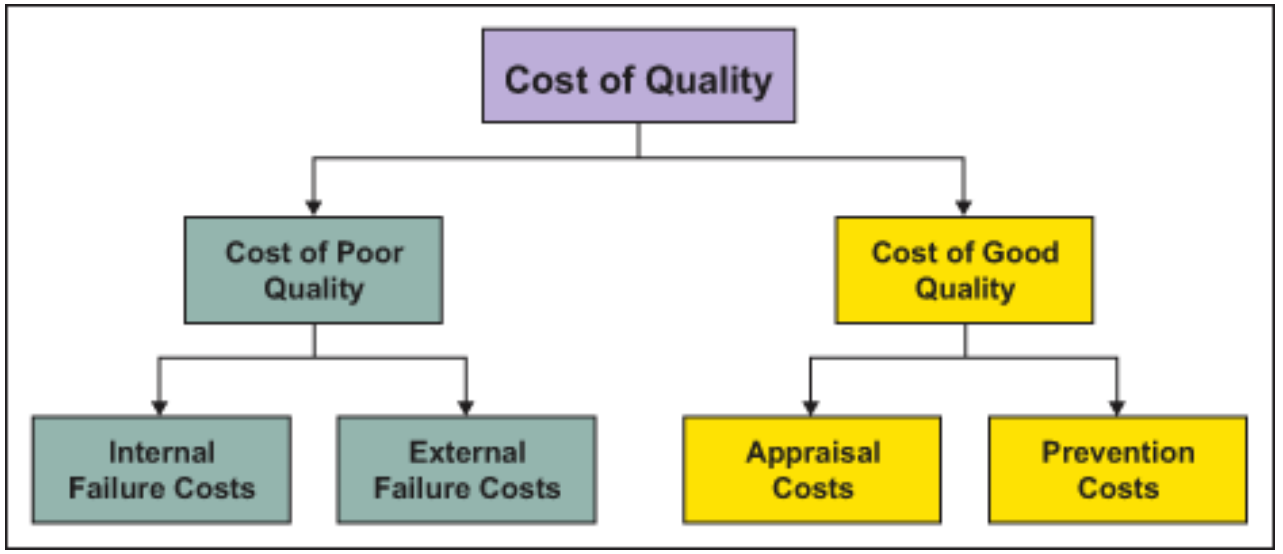
Support. Kan kundernas användning och problem understödjas?
- Identifierbar: det är lätt att identifiera delar av produkten och deras version, eller specifika fel.
- Diagnostisering: är det möjligt att få reda på detaljer kring kundsituationer?
- Undersökningsbar: är det lätt att sätta fingret på fel (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

Cost of Quality - SixSigma

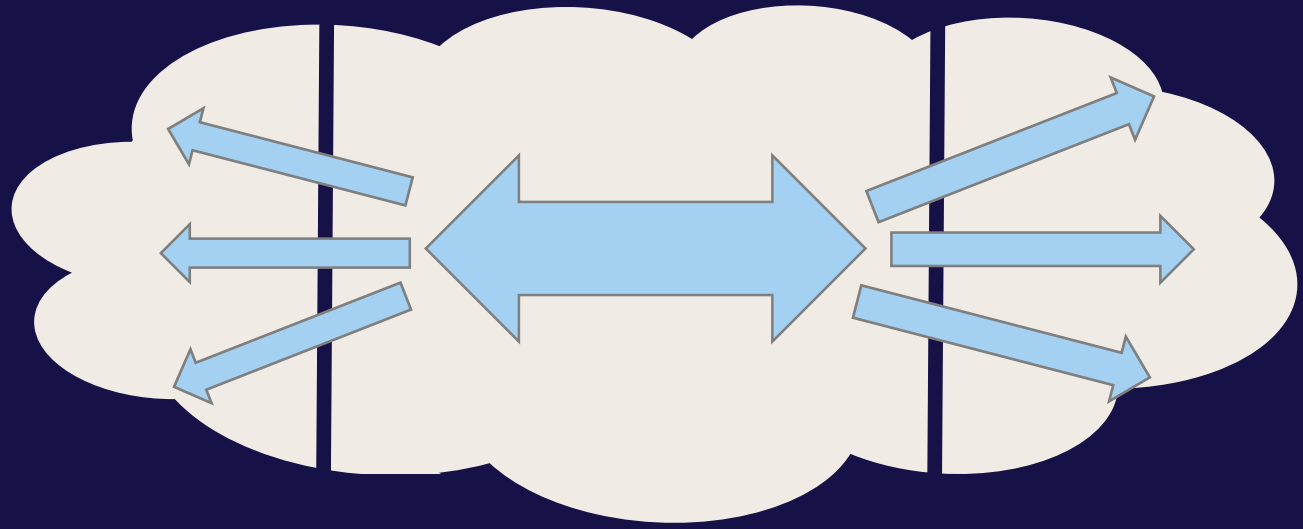


Testningens verkliga uppgift

- 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

Agile



NAP

Control

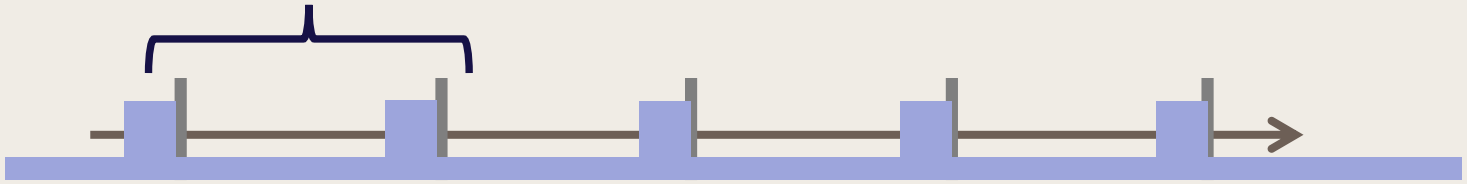
- Visibility = ☺
- Steering = 😐

Kod är en kostnad.

Leverera bevisad funktionalitet.

Hur fel kan det ha hunnit bli?

Minsta möjliga tid till värde



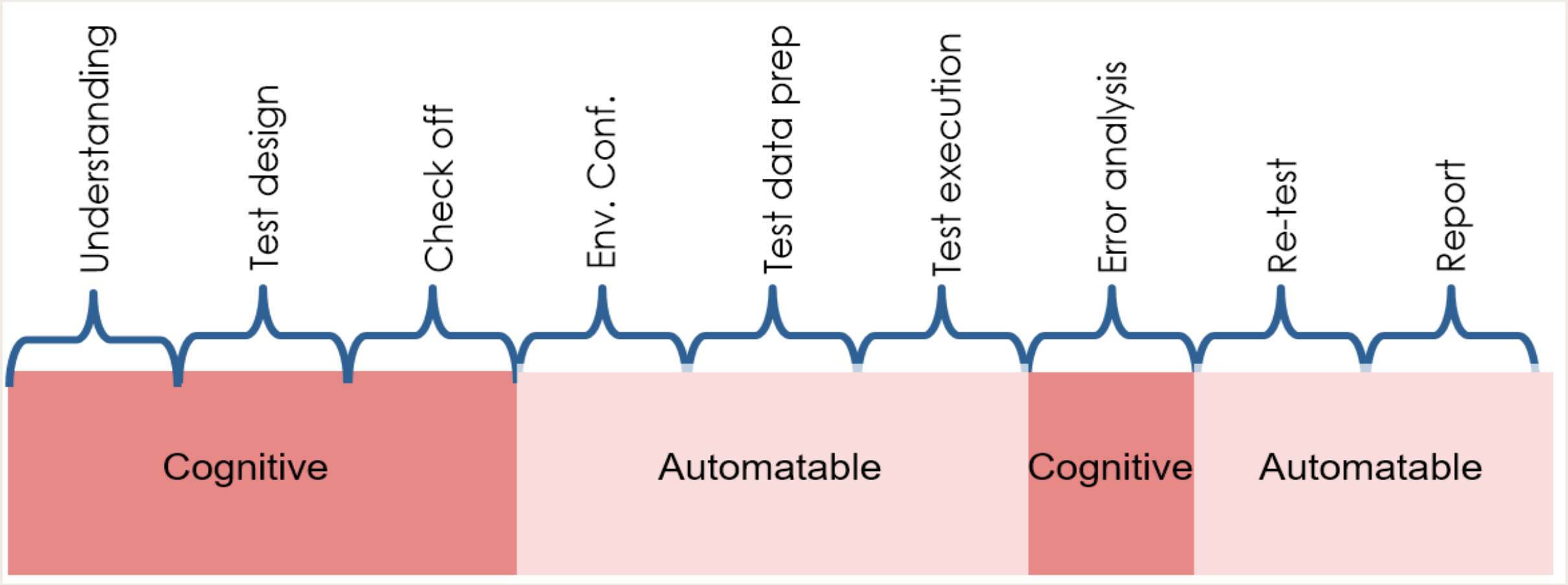
Minsta möjliga tid till värde

Hur fel kan det ha hunnit bli?

[au-tu-ma-tiˊsé-raɵ]

Begrepp

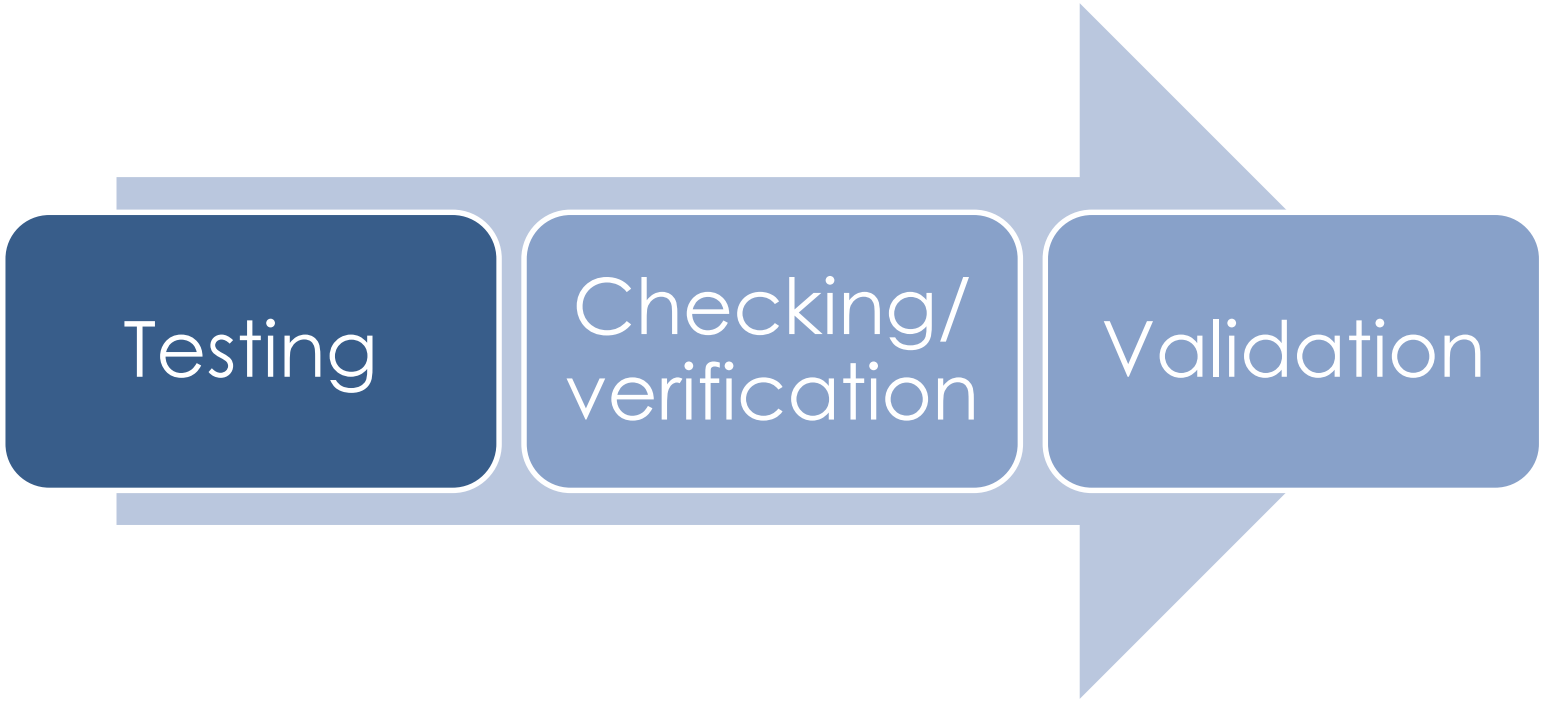
Term	Förklaring
Regressionstest	Tester som görs för att säkra att ingenting gått sönder när man infört en ändring. Testar i hela systemet, inte bara runt ändrade funktionaliteten.
Omtest	Testfall som körs igen, t.ex. vid felrättning.
Utforskande testning	Tester där man dokumenterar testfallen medan man 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	Lim-kod 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


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

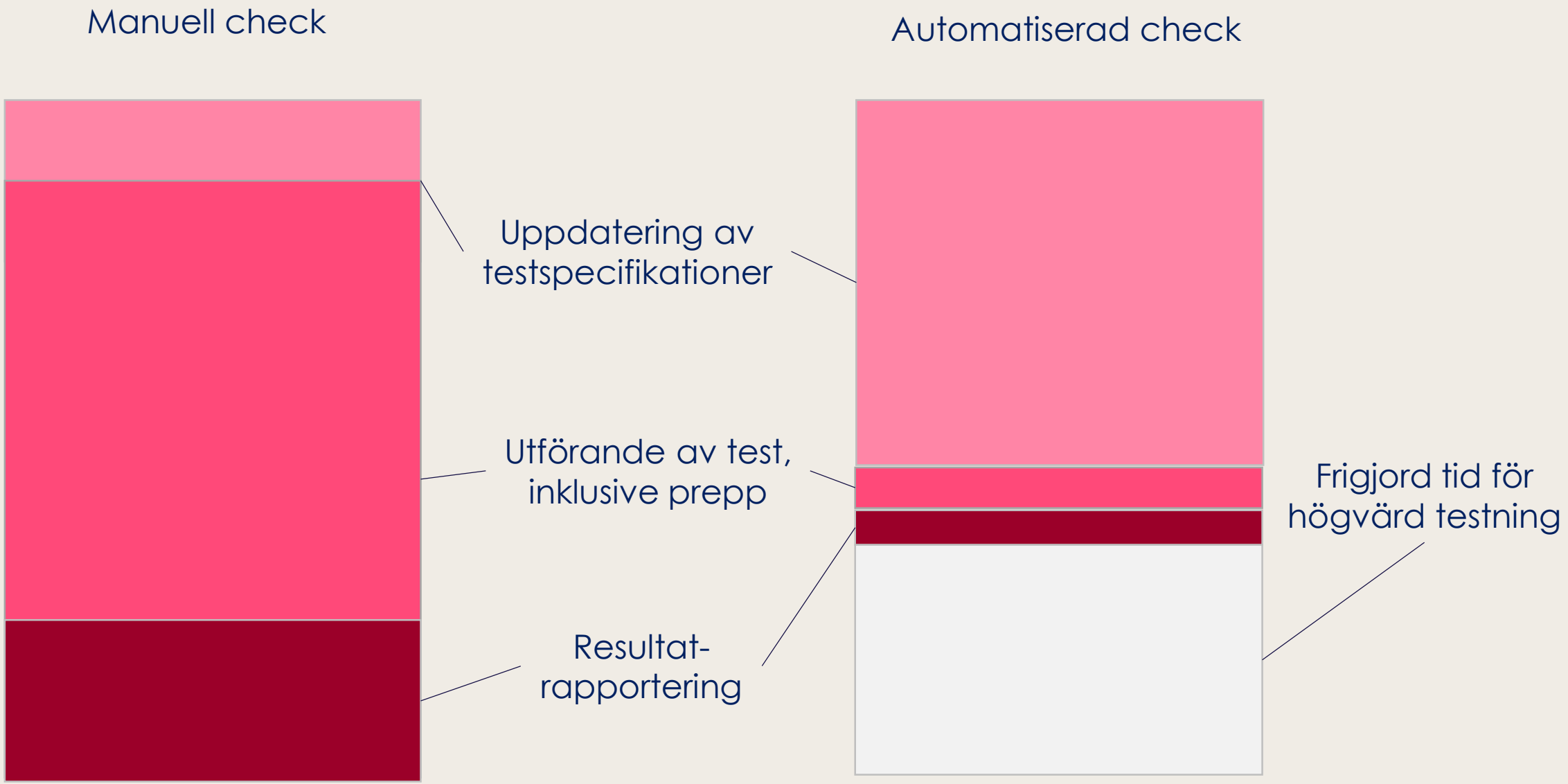
Problem: Närhets-bias

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

Allt annat bör automatiseras - om det går.


Testing
Checking/ verification
Validation

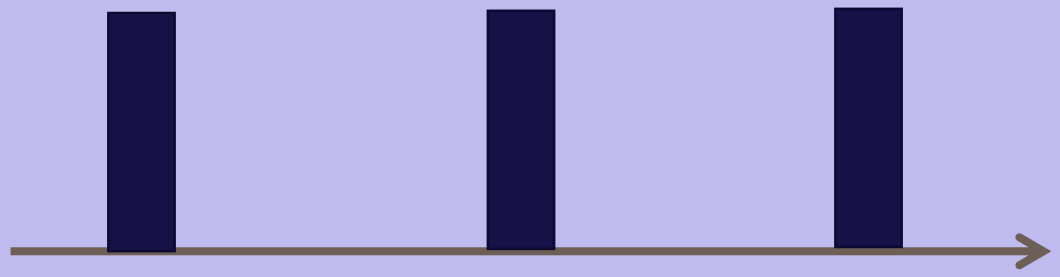
Manuellt arbete vid manuell checkning vs automatiserad checkning



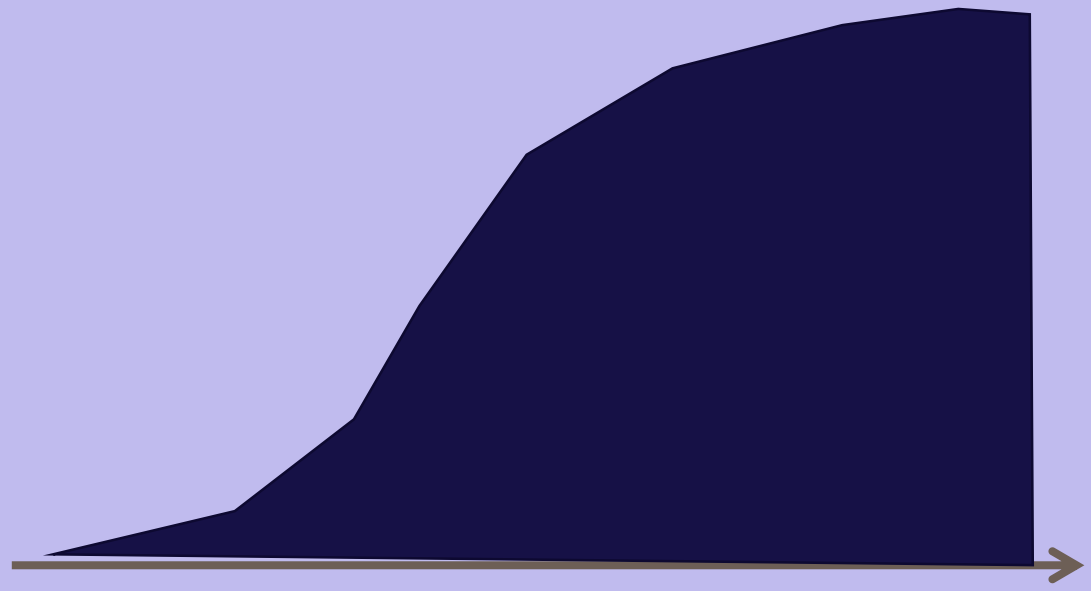
Manuell checkning vs automatiserad checkning

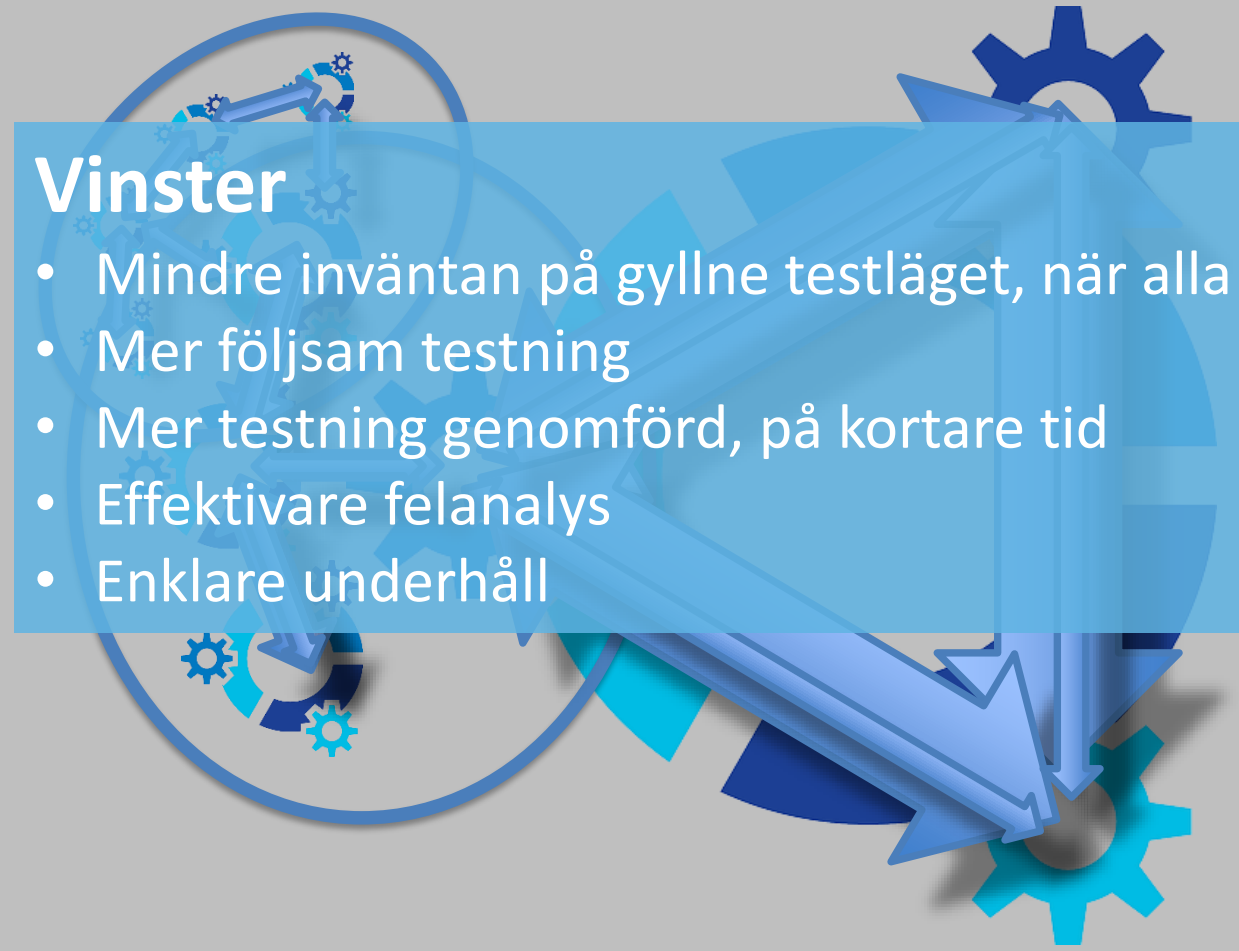
Antal utförda testfall över tid

Manuell testning



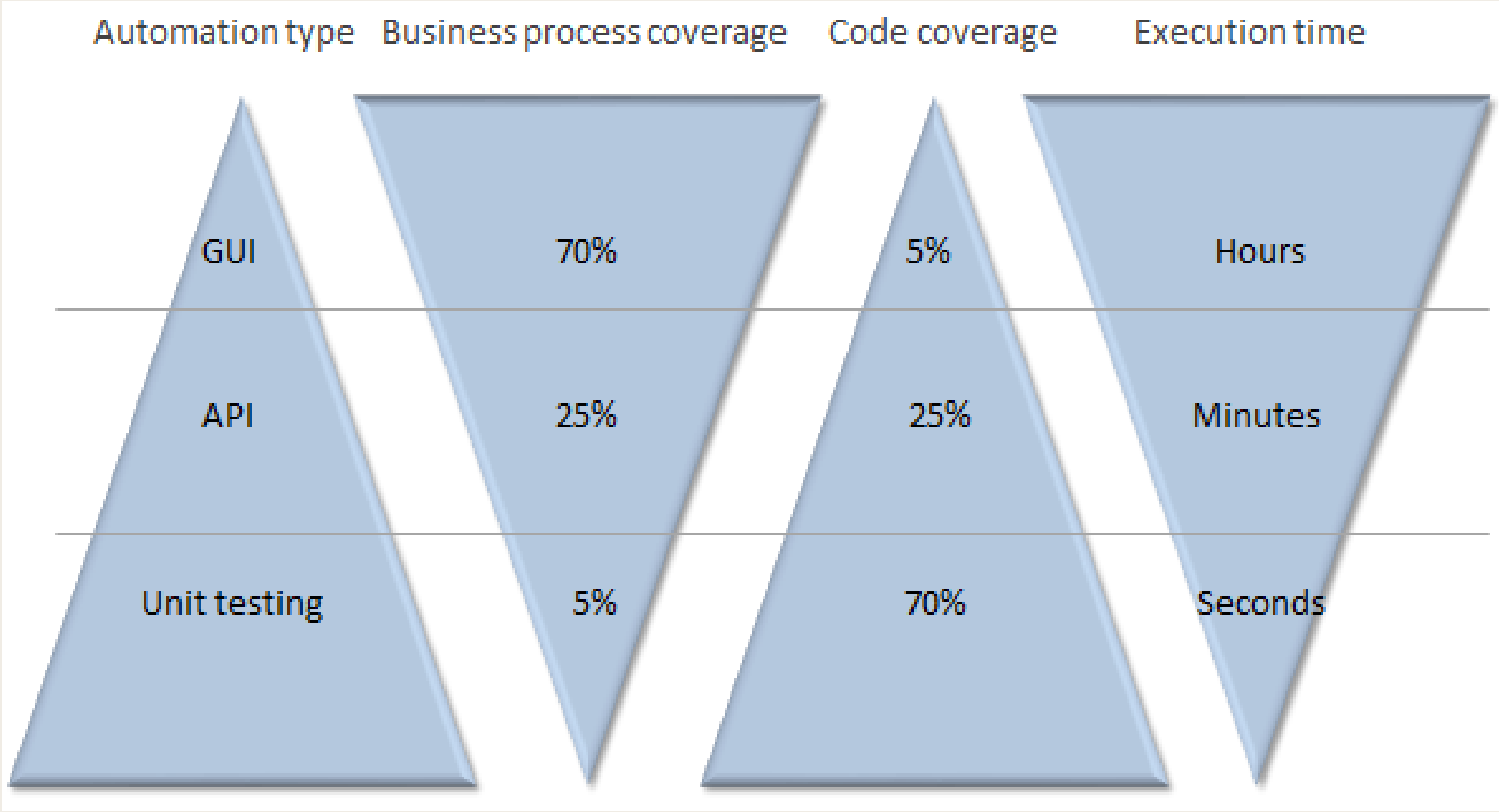
Automatiserad testning





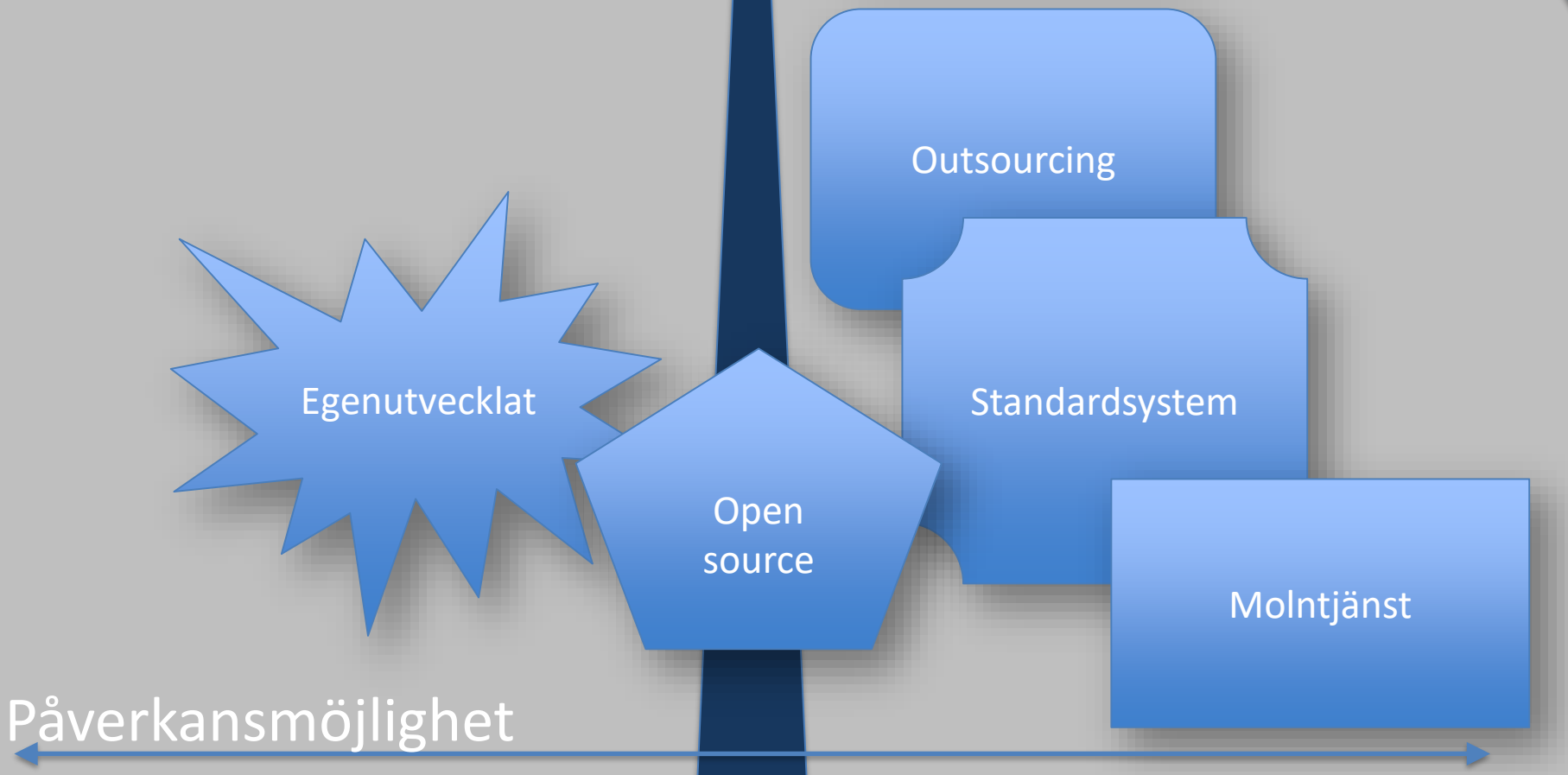
Vinster

- 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


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

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

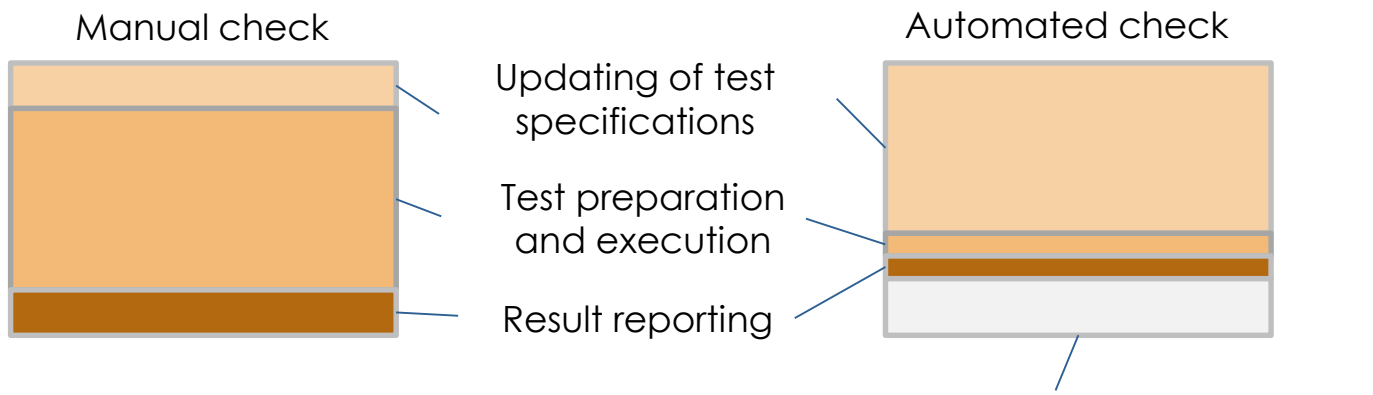
Avidentifiering
Sub-setting
Up-stream
Avtal
UAT
SV



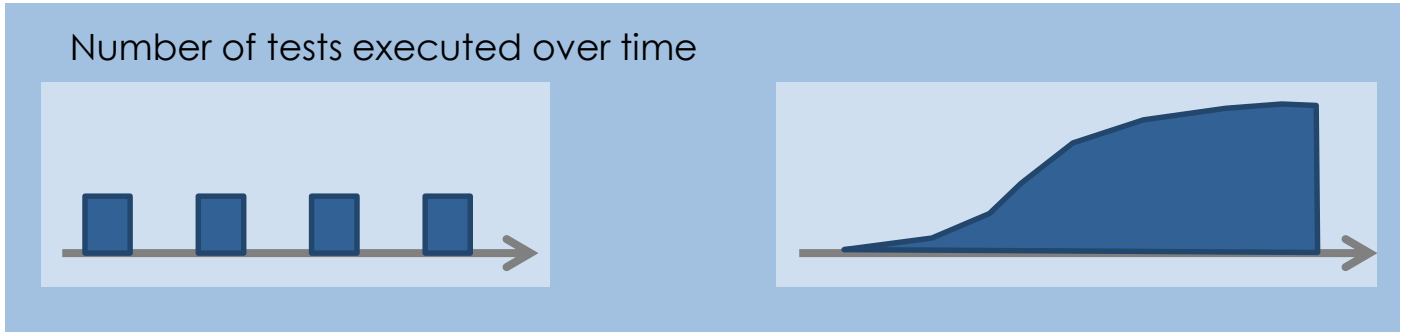
Manual checkning vs automated checkning

Better testing

Faster testing

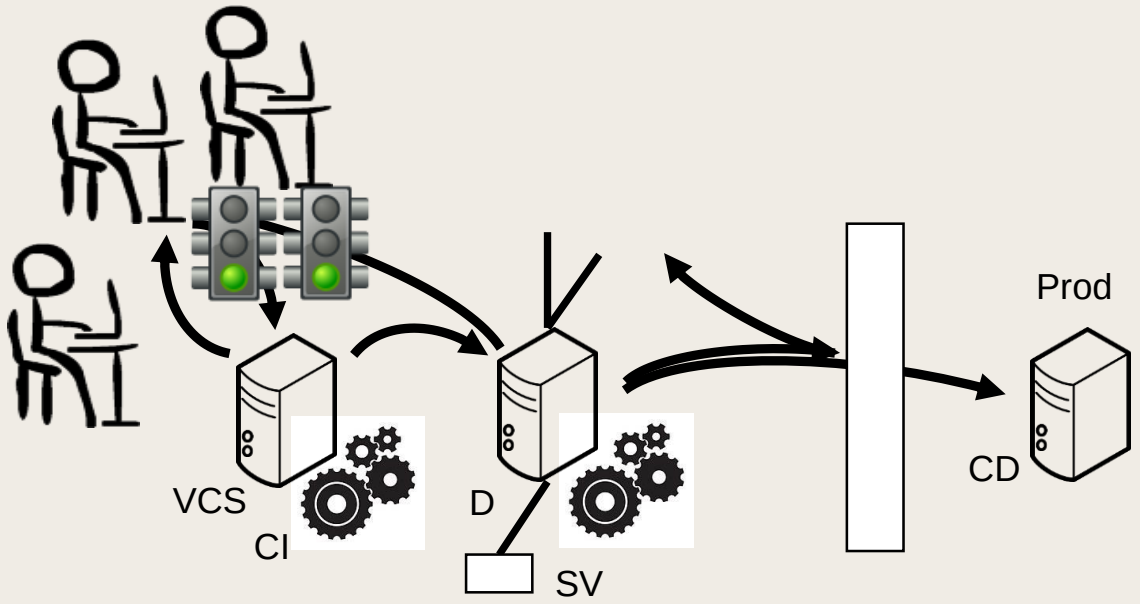


Doesn't save manual time

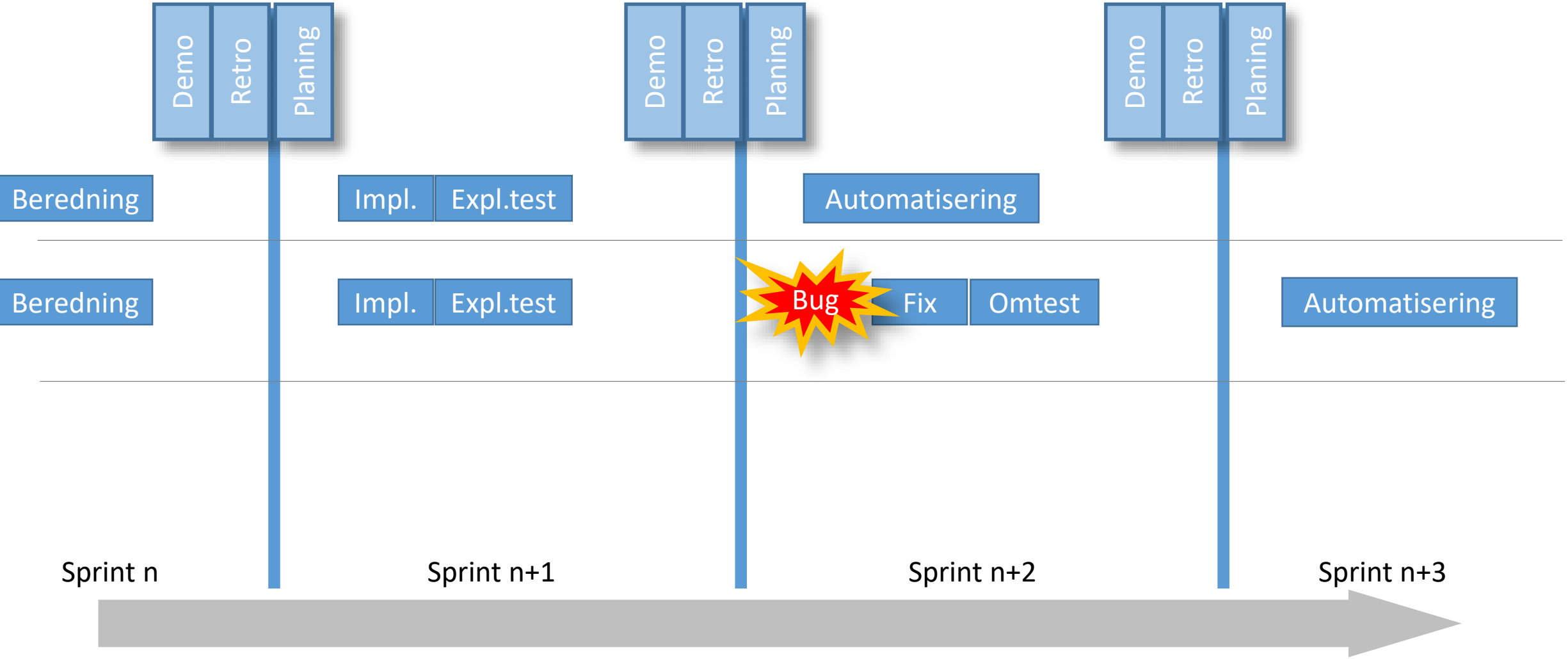


Continuous Delivery

Continuous Integration
Continuous Deploy



Graf över när i tiden…



Det största hotet mot testautomatisering är att det är så kul

Testautomatiseringutmaningar

- Änd
- Änd
- Änd
- Timi
- Defe acerat

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

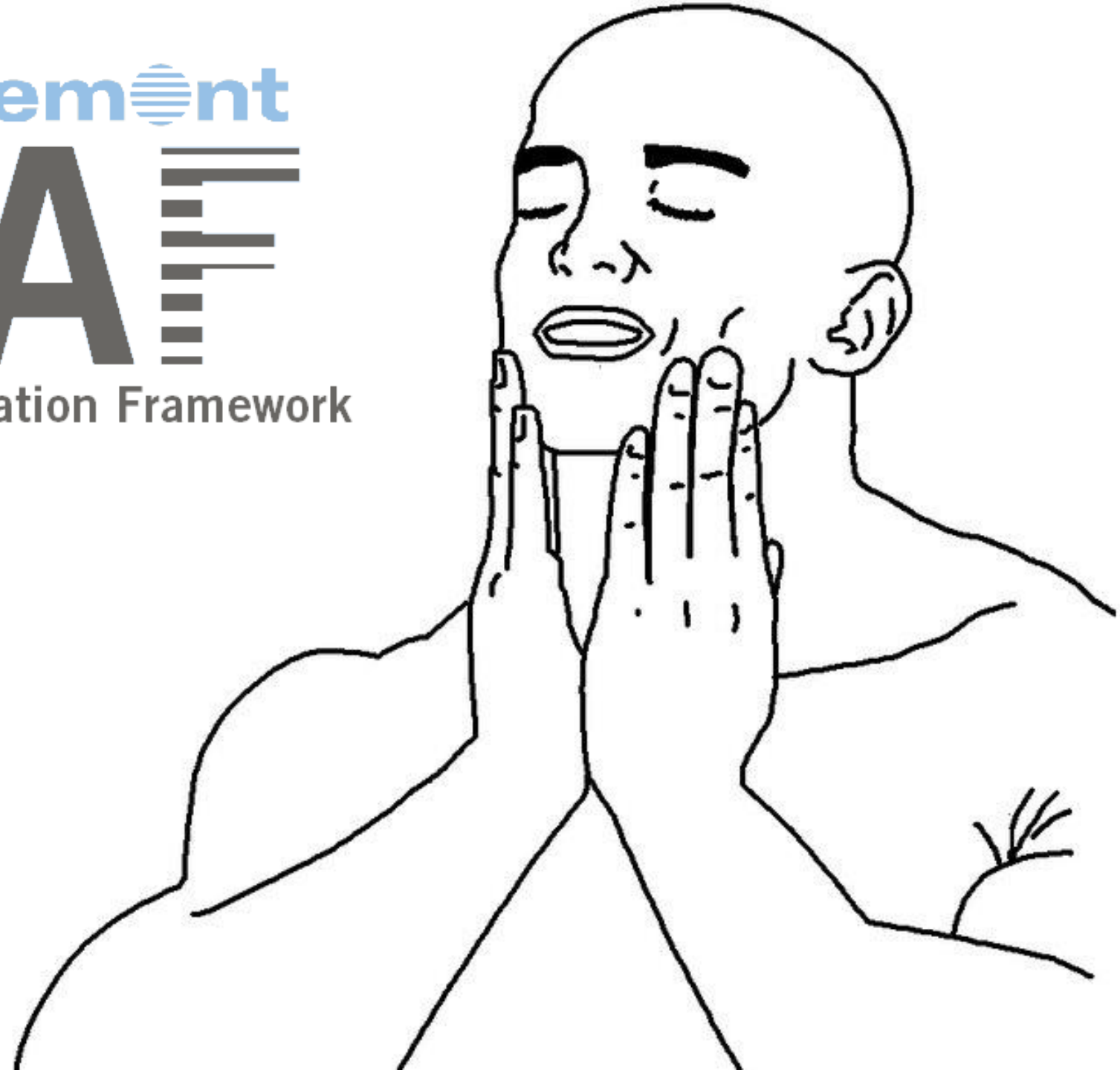
- Ana
- Ändra testfall och applikationsbeskrivningar tar tid
- Implementera ramverk tar tid

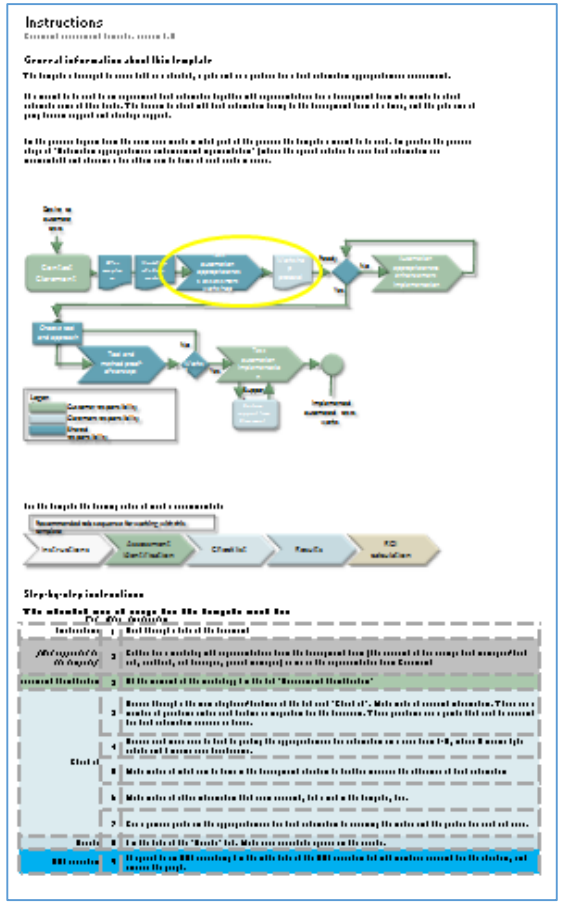
- En person som ska vara sin egen arkitekt för en kodbas med mer beroenden till andra system än något annat
- En person som ska kunna kommunicera med både verksamhet, testare, arkitekter och utvecklare
- En person som ska kunna vara självstyrande och planera sitt eget arbete
- En person som ska ha förståelse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

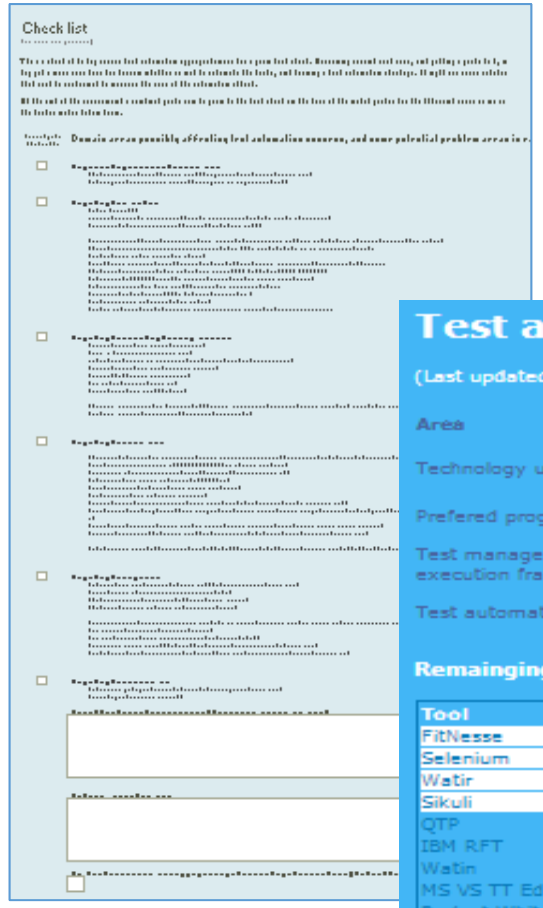
Claremont

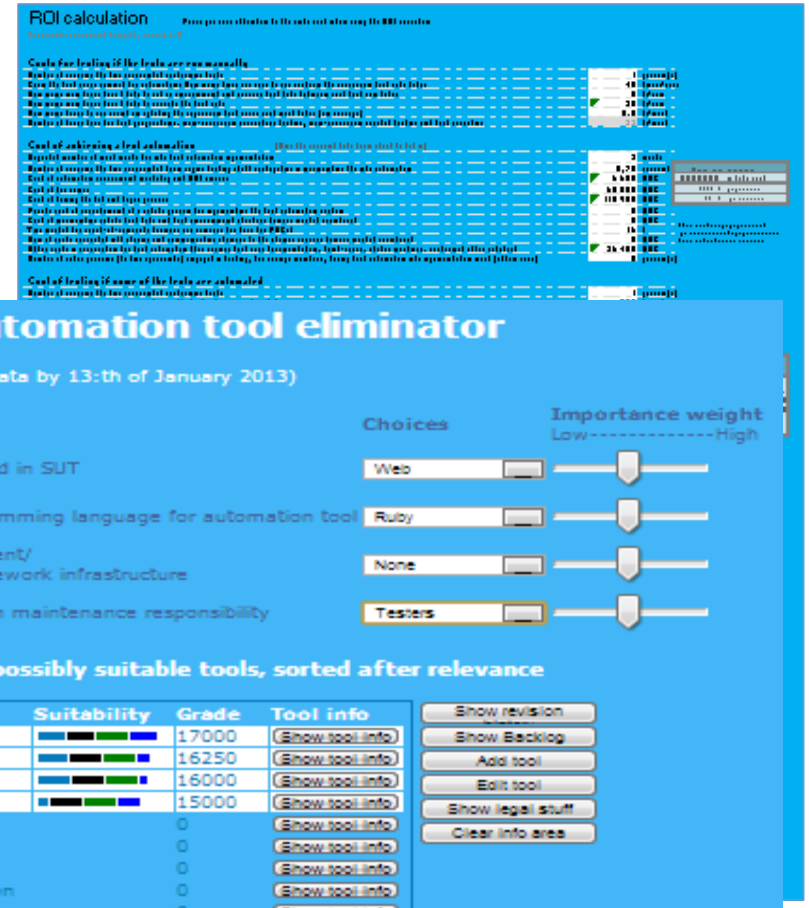
TAF

Test Automation Framework









Test automation tool eliminator

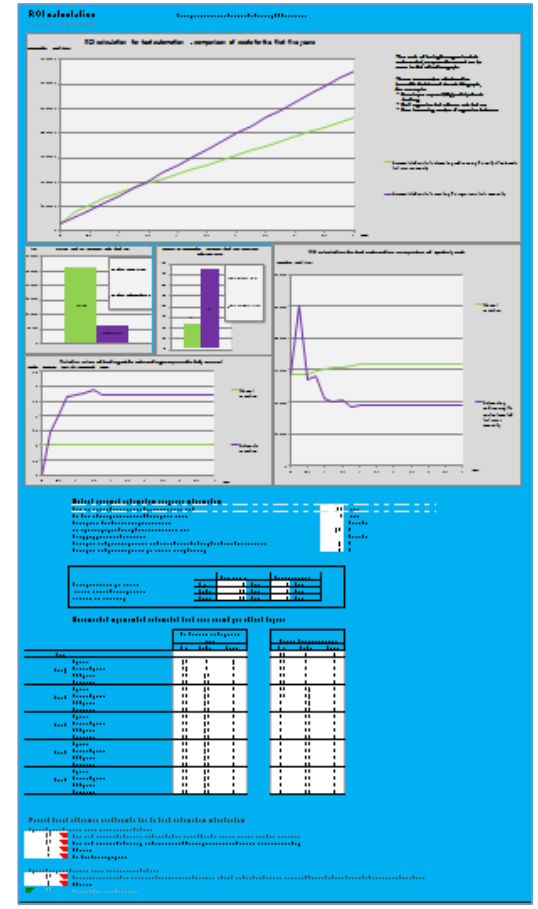
(Last updated data by 13:th of January 2013)

Area	Choices	Importance weight Low----------High
Technology used in SUT	Web	
Prefered programming language for automation tool	Ruby	
Test management/ execution framework infrastructure	None	
Test automation maintenance responsibility	Testers	

Remainging possibly suitable tools, sorted after relevance

Tool	Suitability	Grade	Tool info
FitNesse		17000	Show tool info
Selenium		16250	Show tool info
Watir		16000	Show tool info
Sikuli		15000	Show tool info
QTP		0	Show tool info
IBM RFT		0	Show tool info
Watin		0	Show tool info
MS VS TT Edition		0	Show tool info
Project White		0	Show tool info
Soap UI		0	Show tool info
WebFT		0	Show tool info
TestComplete		0	Show tool info
SilkTest		0	Show tool info
WorkSoft Certify		0	Show tool info
TAO		0	Show tool info
TestPartner		0	Show tool info
Canoo WebTest		0	Show tool info
HTTPunit		0	Show tool info
HTMLunit		0	Show tool info
e-valid		0	Show tool info
Sahi		0	Show tool info
Telerik		0	Show tool info
Windmill		0	Show tool info
QF-Test		0	Show tool info
EggPlant		0	Show tool info
TRICENTIS TOSCA		0	Show tool info

Show revision
Show Backlog
Add tool
Edit tool
Show legal stuff
Clear info area



Varför är TAF skapat?

- Vi ska som konsulter kunna ge större värde för våra kunder
- Våra säljare ska kunna ha något att prata om
- Allt för många testautomatiseringar dör bort med tiden. Det håller inte om vi ska ha egna åtaganden.
- Återanvändning av varandras ansträngning och erfarenheter

Vad är TAF?

Ramverk för testautomatisering

Maven™

Snabb och robust scriptning

Loggning

Kundanpassning

Rapportering

Parameterhantering

Open source

CI/CD-skapat

Java-baserat

Komponentuppdelat

Developers

OkHttp Library

JUnit

Tankar och koncept

- Ingen testautomatisering är bättre än dess rapporter.
- Ta bort störkällor som ger falska fel. Något som är felmarkerat ska vara något att engagera sig i.
- Verktygslåda: Robusta metoder ska finnas redo för det man vill.
- Redo för CI/CD out-of-the-box
- Framtidssäkrat: Verktygsval är trendkänsliga.
- Komplexiteten långt bort från testfallen
- Inget underliggande är oåtkomligt

Vad kan TAF?

JAutomate

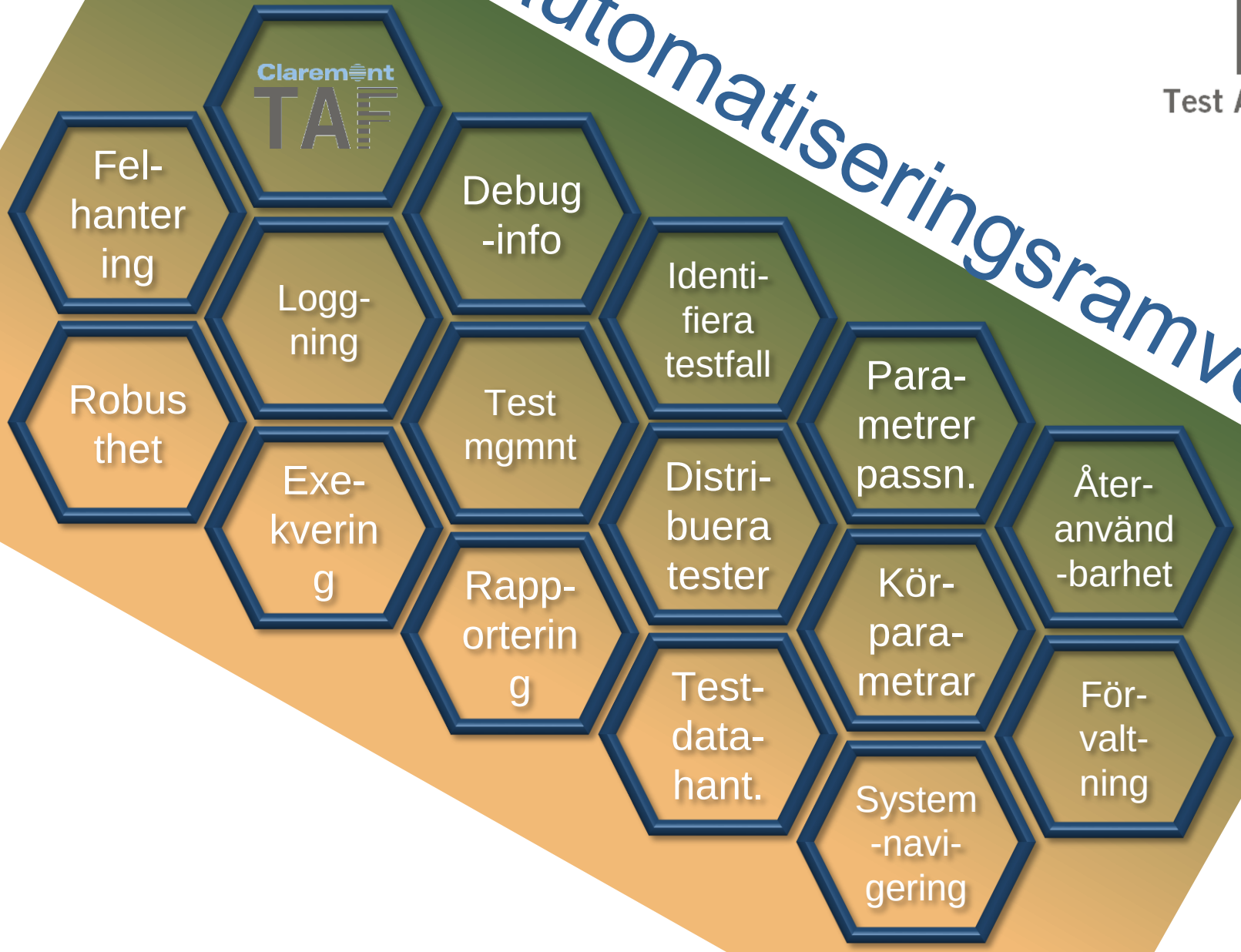
Krav

- Ge stöd
  - Hantera timing-problem
  - Generisk lösning som tål teknikskiften
  - Oavsett förändring ska man bara behöva uppdatera på ett ställe
  - Hantera olika miljöer med samma testfall
  - Rapportera relevanta saker – utan att det drunknar i information
  - Vara enkelt att komma igång med

- Funka varifrån som helst:
  - från CI/CD,
  - och IDE,
  - och för sig självt,
  - och i byggprocesser

Testautomatiseringsramverk

Claremont TAF
Test Automation Framework



Vad TAF ger

- Robust testexkvering
- Framtidssäker automationsinvestering
- Tillrättalagd rapportering
- Skalbarhet
- Mångsidig användning
- Samma exekvering mot olika testmiljöer
- Försprång i implementation
- Eget mandat över testautomatiseringslösningen
- Utbyggbarhet och anpassningsbarhet

”Lösningar”

- Ingen interaktion direkt mot driver
  - Om man byter driver ska man inte behöva uppdatera testfallen
  - Chans att vara bättre än drivern
  - Förberedda funktioner
    - Scriptstyrning
    - Framdriftsteg
    - Verifieringsmetoder
- Gör aldrig underliggande teknologi oåtkomlig
- Pressa bort komplexitet från användaren
- Hantera sann black-box
- Rapportering på flera kanaler
- Undertryck irrelevant information
- Eget loggningsramverk
  - Egna loggnivåer
  - Särskilj kända fel och icke-kända fel

Claremont

TAF

Test Automation Framework

Debug
Info

Executed
Execution problem

Verification passed
Verification failed
Verification problem

Framework error

Anpassade loggnivåer

- I rapporter grupperas liknande fel
- Separat hantering av kända fel jämfört med nya fel
- Extra hjälpinformation med vid avvikelser
  - Screenshots, med gul ruta runt objektet som interageras med
  - Desktop screenshot
  - Startade och stoppade processer sedan testfallsstart
  - Datornamn, user, CPU- och minnesinformation, OS information o.s.v.
  - Tydlig loggning – med förslag och tips där det går

Exekveringshantering

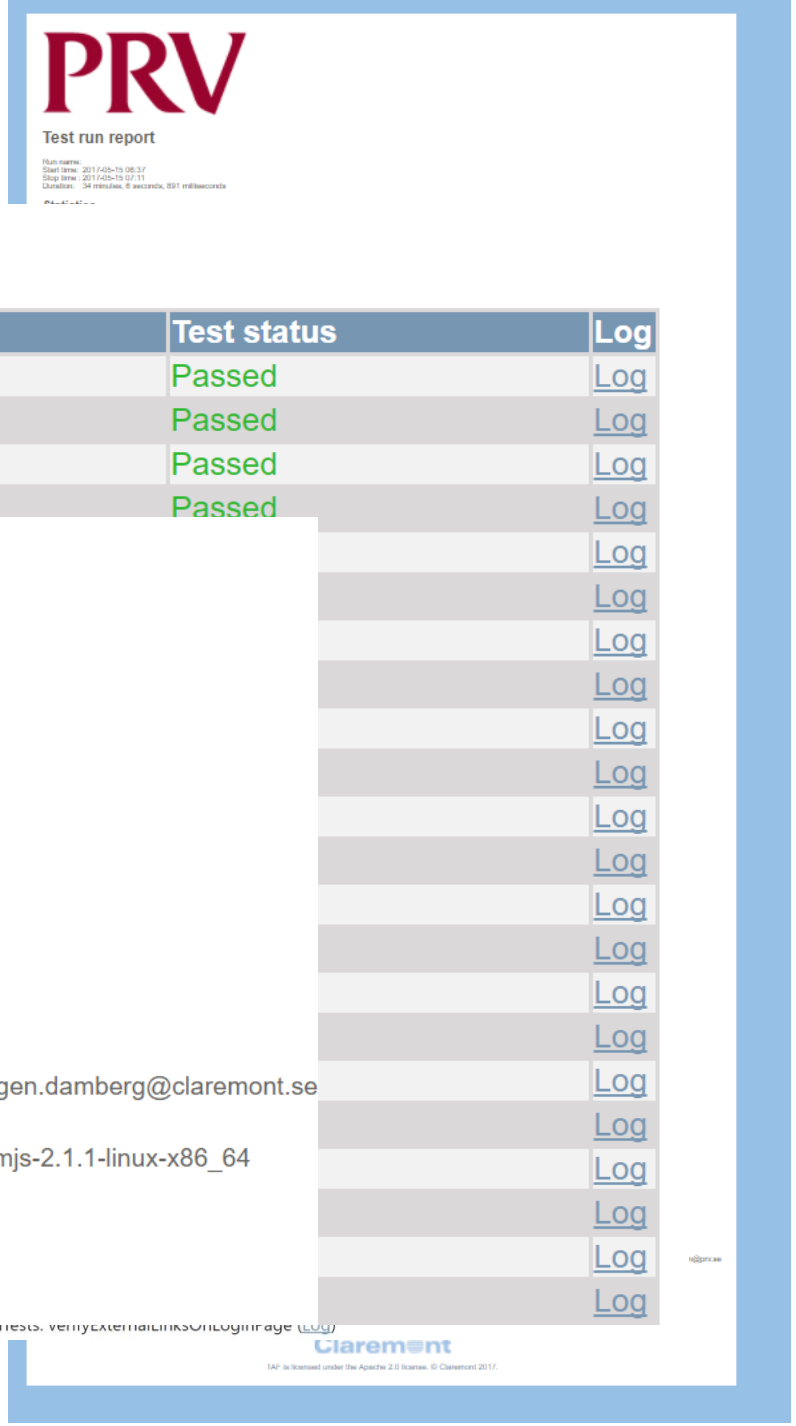
- Command line interface (CI/CD, Jenkins, Team City)
- IDE (Eclipse, IntelliJ o.s.v.)
- JUnit Runner Console

Delar i TAF

- TAF Core
- TAF Backend Server
- TAF Testlink Adapter Server

- TAF sample project

Rapportering


PRV

Test run report

Test case summary

Test set	Test case name	Test status	Log
se.prv.tests.rest.contractserver.ContractServiceTest	approveUserContract	Passed	Log
se.prv.tests.rest.contractserver.ContractServiceTest	getApprovedUserContract	Passed	Log
se.prv.tests.rest.contractserver.ContractServiceTest	getContracts	Passed	Log
se.prv.tests.rest.contractserver.ContractServiceTest	getActualUserContract	Passed	Log
se.prv.tests.r			Log
se.prv.tests.r			Log
se.prv.tests.r			Log
se.prv.tests.r			Log
se.prv.tests.r			Log
se.prv.tests.r			Log
se.prv.tests.r			Log
se.prv.tests.r			Log
se.prv.tests.r			Log
se.prv.tests.r			Log
se.prv.tests.r			Log
se.prv.tests.r			Log
se.prv.tests.r			Log
se.prv.tests.r			Log
se.prv.tests.r			Log
se.prv.tests.r			Log

Registered known errors not encountered

['Service is down.']
["Anförda' links does not work properly']

Test run settings

Email sender address	jorgen.damberg@gmail.com
Run settings file	/var/lib/jenkins/workspace/AutoTest MyPages/TAF/runSettings.properties
Access path to files	file:///var/lib/jenkins/workspace/AutoTest MyPages/TAF/
HTML reports link prefix	http
Report email recipients	shamsul.khan@claremont.se,per.danielsson@prv.se,karin.jansdotter@prv.se,jorgen.damberg@claremont.se
Report log folder	smb://172.16.202.10/autotest
Path to PhantomJS binary	/var/lib/jenkins/workspace/AutoTest MyPages/src/main/resources/drivers/phantomjs-2.1.1-linux-x86_64
Path to Firefox browser binary	/usr/bin/firefox
URL to TAF Backend Server	http://172.17.24.168:80/taf
URL to TAF Testlink Adapter Server	http://-server-url-not-set-/taftestlinkadapter
TAF version	2.6.0

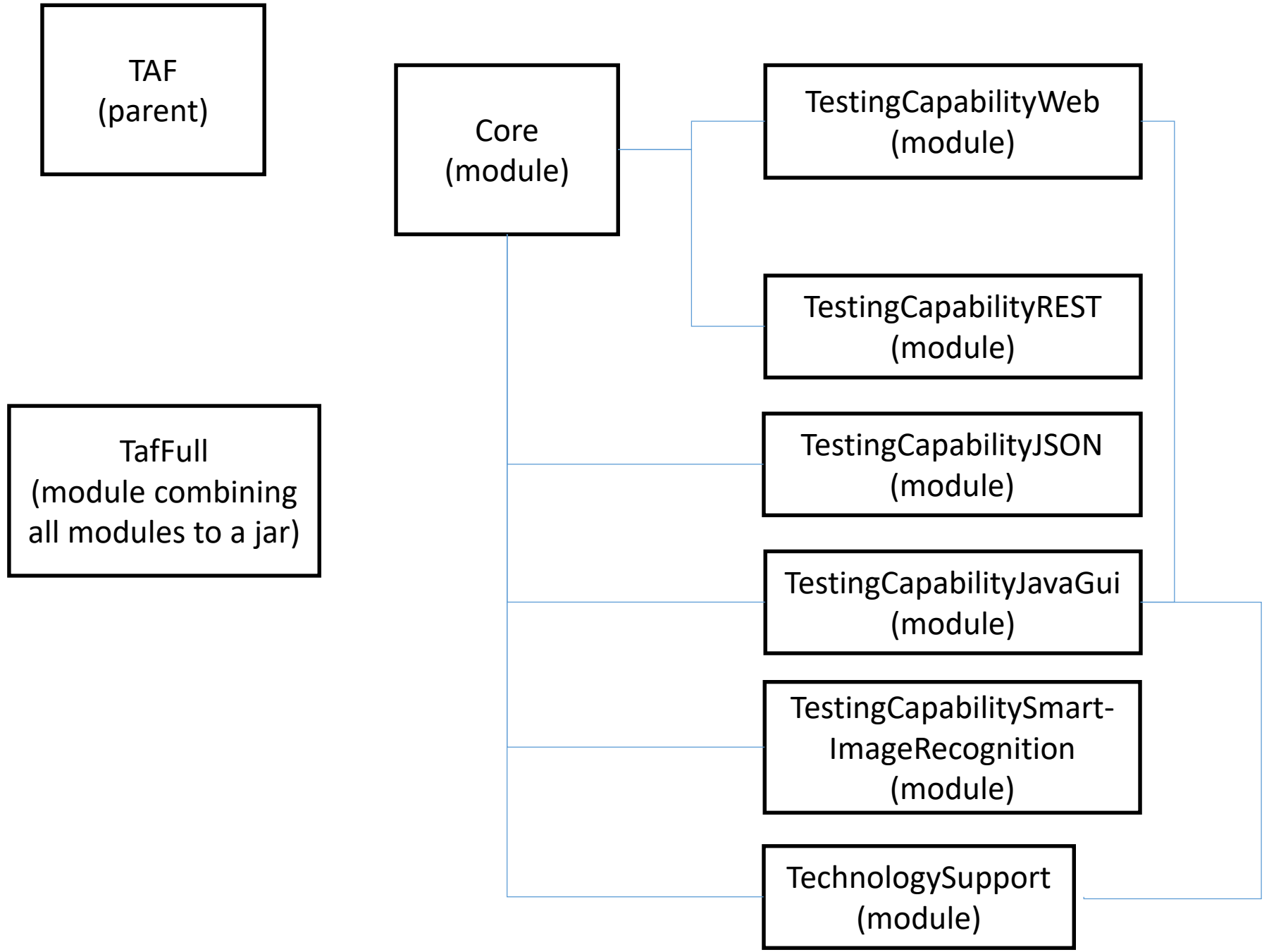
Claremont

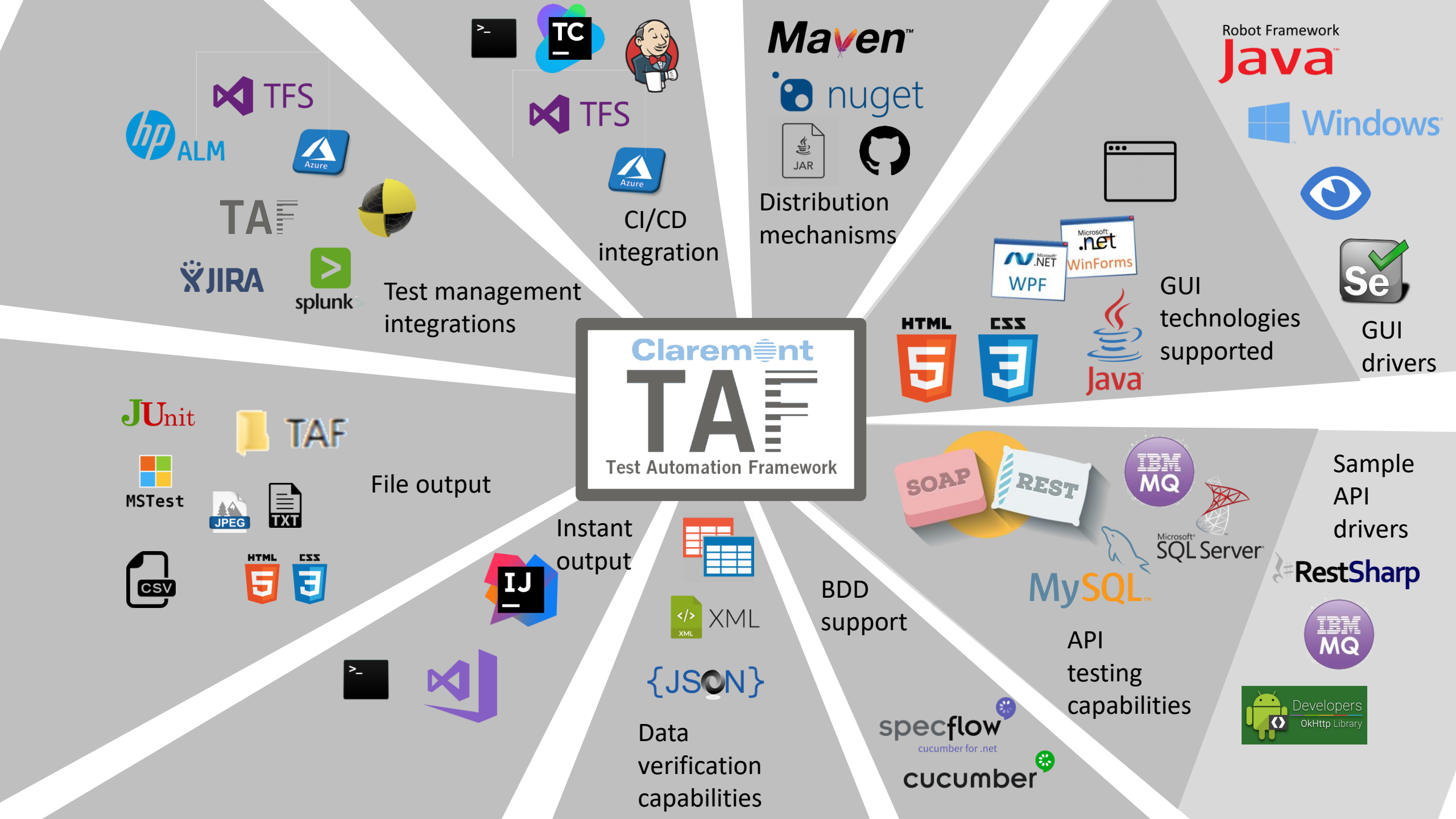
Statistics

YIPPIE!

Result	Count
Passed test cases	79

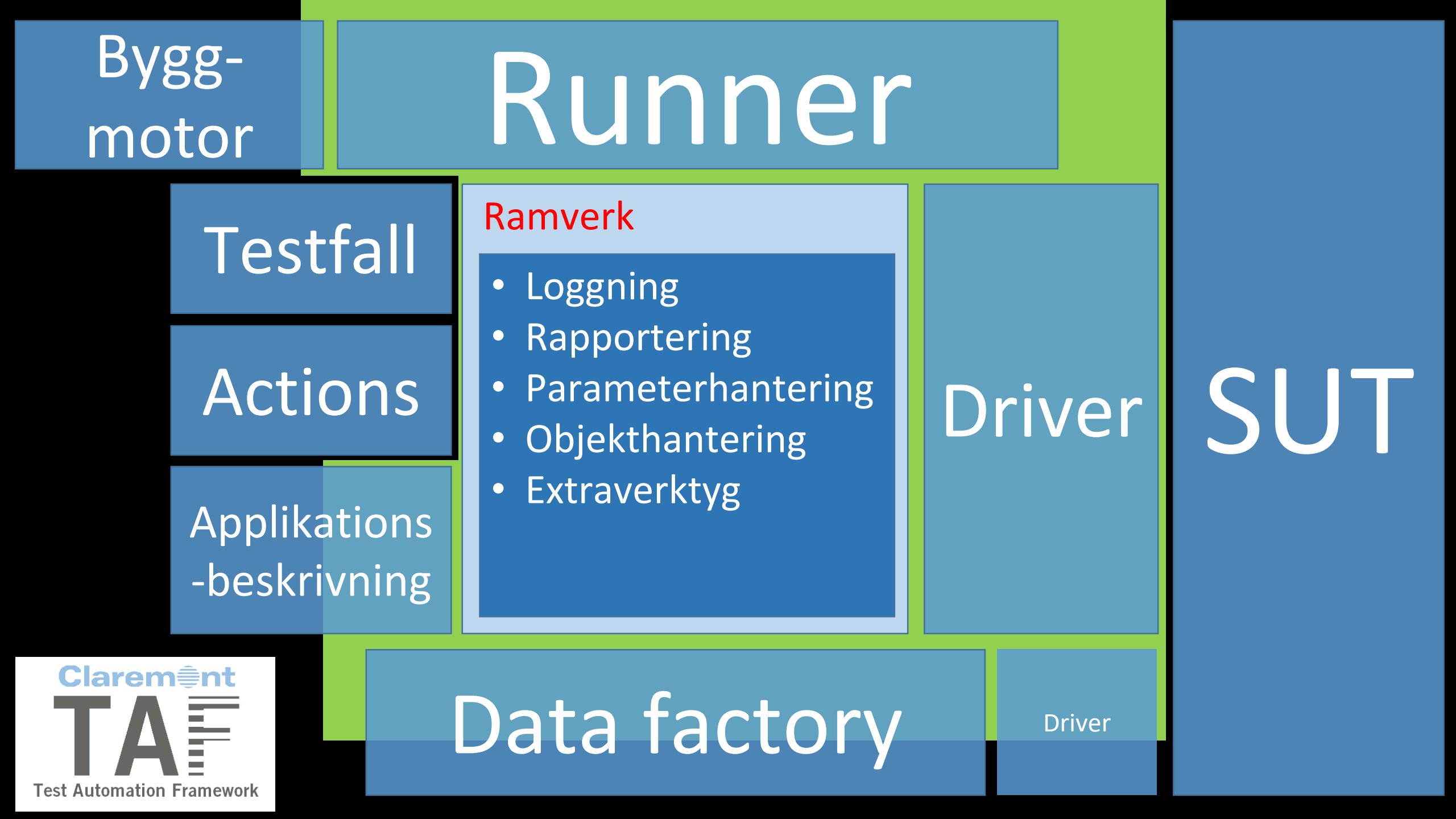
open source


TAF
(parent)
Core
(module)
TestingCapabilityWeb
(module)
TestingCapabilityREST
(module)
TestingCapabilityJSON
(module)
TestingCapabilityJavaGui
(module)
TestingCapabilitySmart-
ImageRecognition
(module)
TechnologySupport
(module)
TafFull
(module combining
all modules to a jar)


Claremont
TAF
Test Automation Framework
Test management integrations
TFS
ALM
Azure
TAF
JIRA
splunk
CI/CD integration
TC
TFS
Azure
Distribution mechanisms
Maven
nuget
JAR
GUI technologies supported
WPF
WinForms
HTML
CSS
Java
GUI drivers
Robot Framework
Java
Windows
Sample API drivers
RestSharp
Developers
OkHttp Library
API testing capabilities
SOAP
REST
IBM MQ
Microsoft SQL Server
MySQL
BDD support
specflow
cucumber for .net
cucumber
Data verification capabilities
XML
{JSON}
Instant output
IJ
File output
JUnit
TAF
MSTest
JPEG
TXT
CSV
HTML
CSS

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


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

Jenkins

Te

Plugins

- TFS-integration
- Testlink-integration
- Splunk-integration
- Video-logging
- mm

-beskrivning

ring

Broken link checker
Browser console checker

Java

Developers
OkHttp Library

Webb
REST
Java

Genererat data, framsökt data, konstruerat data

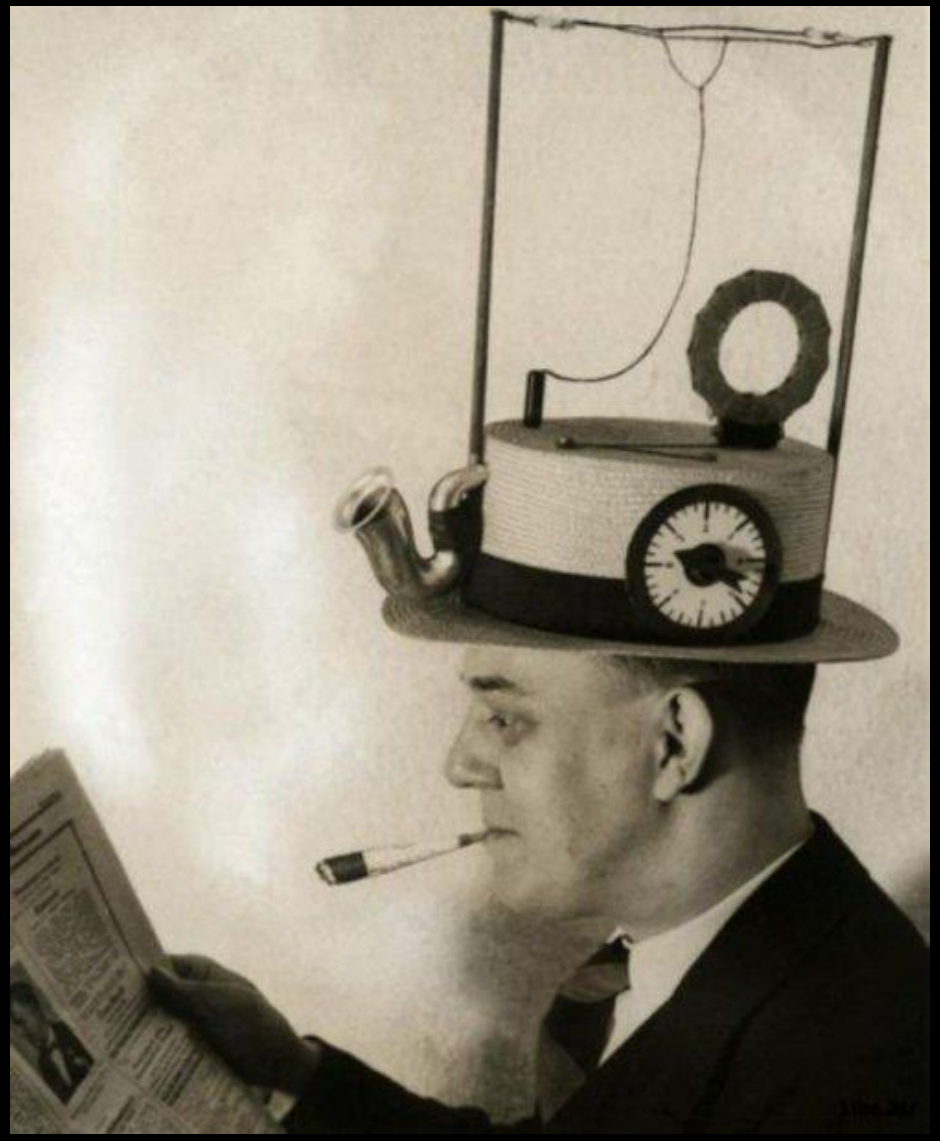
MySQL

Claremont
TAF
Test Automation Framework

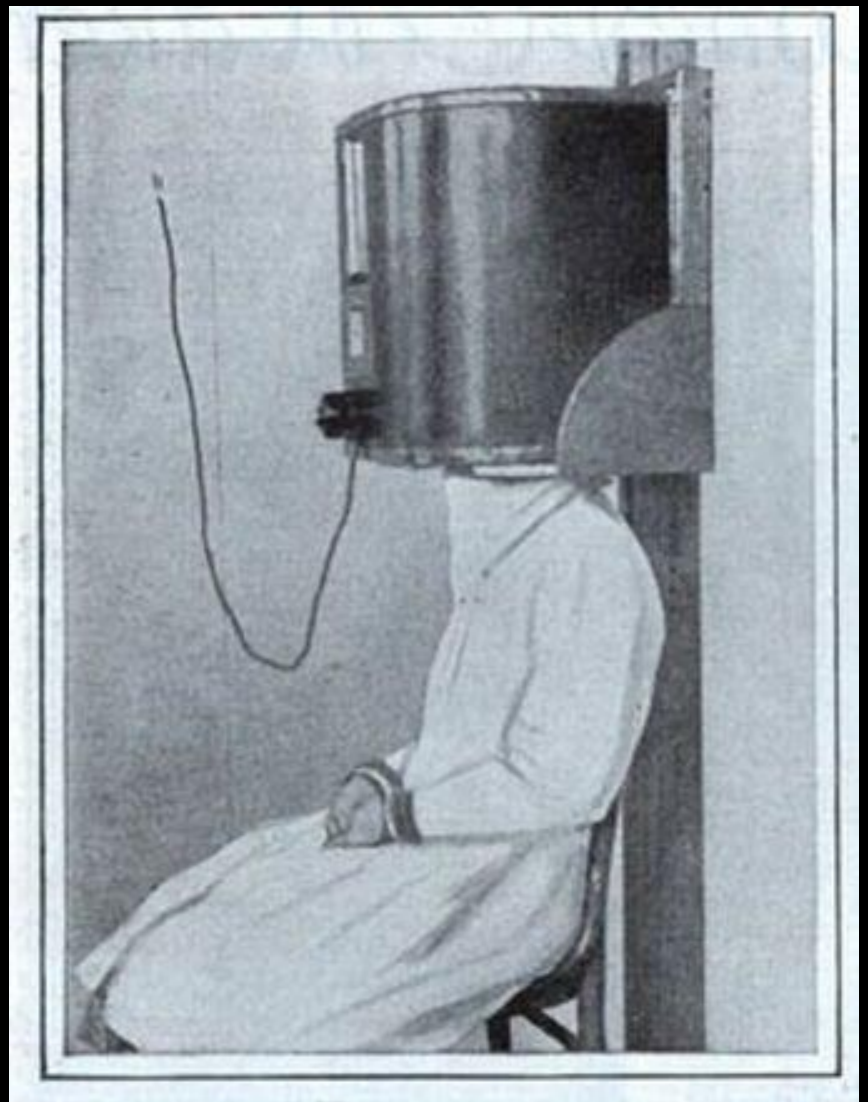

DO NOT CROSS
DO NOT CROSS

”Kunden” – 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



Icke-kunden – Kodande utvecklare



- 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
- Ovana vid att hantera alla utmaningar som testare möts av
  - De ger upp och tycker att automatisering är jobbigt

Claremont

Mode
Keywo
Datad
Specif

	A
1	Name
2	John Smith J
3	Clare Jeffers
4	Susan McLar
5	Charles Dodg
6	Steve Johns
7	Samuel Clem
8	Bob Feather
9	Mark Smith

```
page
ea
ast' tea
y cart

 in my cart



 shipping
```

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(,)

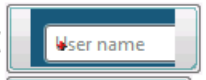
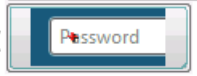
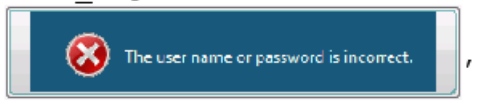
Keyboard Actions

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

Produktutveckling som open-source

Pros

- Konsumentprodukt: Många att tillfredställa – arkitekturellt jobbigt
- Tvärfunktionella insikter
- Bra diskussioner för att bottna saker

Man blir mycket bättre

Cons

- Konsumentprodukt: Blir snabbt mångas hjälte
- Licensstök
- Mycket diskussioner för att bottna saker

Fåfänga

Utkomst och lärdomar

- Man blir en väldigt mycket bättre kodare av att jobba med open source
- Kommer nu igång på timmar istället för månader
- Varje teams förbättringar kommer alla till gagn
- Vi har något konkret och välkänt att prata testautomatisering kring – vilket gör diskussionerna djupare och oss bättre
- En handfull organisationer använder TAF idag, i olika system

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

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?

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

Länkar

- Huvudsida på GitHub https://github.com/claremontqualitymanagement/TestAutomationFramework
- Wiki: https://github.com/claremontqualitymanagement/TestAutomationFramework/wiki
- Introduktion: http://46.101.193.212/TAF/documentation/TAF%20introduction.html
- Mallprojekt att börja med: https://github.com/claremontqualitymanagement/TemplateProjectForTAF

- TAF Introduktion
- TAF Wiki
- TAF på GitHub