Pruebas de Software: Automatización por Siempre

Gilberto Sánchez Mares

Prácticas modernas para crear software con calidad y sabor

#SGVirtual

SG VIRTUAL CONFERENCE

MEXI TESTER

PRUEBAS DE SOFTWARE

AUTOMATIZACIÓN POR SIEMPRE

CONTENIDO

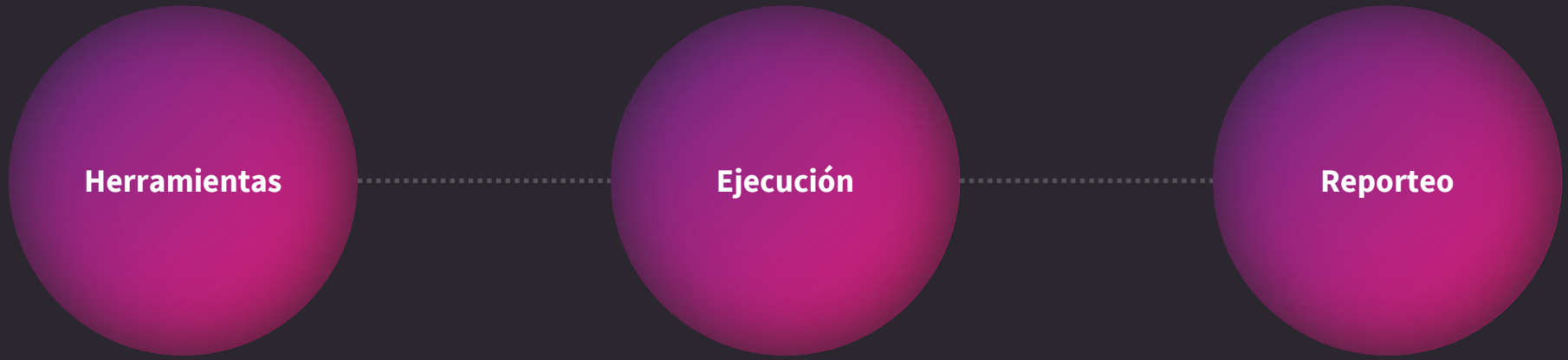
INTRODUCCIÓN

SG VIRTUAL CONFERENCE

“

El objetivo de la automatización de pruebas es **reducir** el número de casos de prueba que se ejecutan manualmente y **no eliminar** la prueba manual.

Tareas en la Automatización



Uso de herramientas de software especiales.

Ejecución de las pruebas.

Comparar los resultados reales con los resultados previstos.

Objetivos

1. Mejora de la eficiencia de las pruebas.

2. Proporcionar una cobertura de funciones más amplia.

3. Reducir el costo total de la prueba.

Objetivos

4. Realizar pruebas que el esfuerzo manual no puede.

5. Acortar el período de ejecución de la prueba.

6. Aumentar/reducir el tiempo requerido para los ciclos de prueba.

Ventajas

Se pueden ejecutar más pruebas por build

Crear pruebas que no se pueden hacer manualmente

Las pruebas pueden ser más complejas

Las pruebas se ejecutan más rápido

Las pruebas están menos sujetas a errores del operador

Uso más efectivo y eficiente de los recursos de prueba

Feedback más rápidos sobre la calidad del software

Mayor confiabilidad y consistencia del sistema

Desventajas

- Costos adicionales involucrados
- Inversión inicial para configurar TAS
- Requiere tecnologías adicionales
- Habilidades de desarrollo y automatización
- Mantenimiento continuo de TAS
- Puede distraer la atención de los objetivos de las pruebas
- Las pruebas pueden volverse más complejas
- Errores adicionales pueden ser introducidos

limitaciones

No todas las pruebas manuales se pueden automatizar

Verifica resultados interpretables por máquina

Dichos resultados deben ser interpretados por un experto

No es un reemplazo para las pruebas exploratorias

Factores de Éxito

Test Automation Architecture (TAA)

Está muy alineada con la arquitectura del producto de software. Debe quedar claro qué requisitos funcionales y no funcionales debe soportar la arquitectura.

SUT Testability

Debe diseñarse para la capacidad de prueba que admita pruebas automatizadas, p. e.:

- Desacoplar la interacción y los datos de la UI.
- API deben exponerse como públicos.

Test Automation Strategy

Una estrategia de automatización de pruebas práctica y consistente que aborda la mantenibilidad y la consistencia del SUT.

Se deben considerar los costos, beneficios y riesgos de aplicarla a diferentes partes del código.

Test Automation Framework (TAF)

Debe ser fácil de usar, bien documentado y mantenible, admitir un enfoque coherente para automatizar pruebas.

Para establecer un fácil uso y mantenibilidad se debe: implementar reportes fácilmente, facilitar la resolución de problemas, documentar los tC automatizados, trazabilidad, fácil despliegue, monitorizar, etc.

PREPARACIÓN

Factores del SUT que Influyen en la Automatización

01

Interfaces

Los casos de prueba automatizados invocan acciones en el SUT. Para esto, el SUT debe proporcionar interfaces a través de las cuales se pueda controlar. Esto se puede hacer a través de los controles de la interfaz de usuario, pero también a través de interfaces de software de nivel inferior.

02

Software de Terceros

A En ocasiones, el SUT no solo consta de software escrito en la organización de origen, sino que también puede incluir software proporcionado por terceros.

03

Niveles de Intrusión

Diferentes enfoques de automatización de pruebas (usando diferentes herramientas) tienen diferentes niveles de intrusión. .

04

Diferentes Arquitecturas

Diferentes arquitecturas SUT pueden requerir diferentes soluciones de automatización de pruebas. Es posible que estas sean manejadas por la misma estrategia de automatización de pruebas, pero eso requiere una estrategia híbrida con la capacidad de admitirlas.

05

Tamaño y Complejidad

Considerar el tamaño y la complejidad del SUT actual y los planes para el desarrollo futuro. Para un SUT pequeño y simple, un enfoque de automatización de pruebas complejo y ultraflexible, puede no estar justificado..

Evaluación y Selección de Herramienta

fuente: https://www.browserstack.com/guide/test-automation-tool-evaluation-checklist

Presupuesto

Las herramientas pueden implicar costos de licencia junto con costos operativos, costos de mantenimiento y, a veces, costos de soporte.

Uso Efectivo

Diseño de scripts de prueba efectivos y elocuentes que garanticen cobertura máxima de prueba es esencial para aprovechar completamente la herramienta.

RoadMap

Cualquier herramienta que se elija debe tener un historial de actualización periódica para admitir nuevas tecnologías, innovaciones y técnicas.

Navegadores, Dispositivos, OS

Las pruebas reales del dispositivo no son negociables si está buscando resultados precisos y confiables.

Tecnología Usada

La herramienta debe caber en su stack tecnológico, no al revés. Antes de realizar una compra, ten muy claro qué otras herramientas se utilizan en tu equipo o empresa.

Requerimientos Técnicos

Soporte de plataforma, soporte de idioma, soporte, usabilidad, reutilización / mantenibilidad de scripts.

Reporteo

Si una prueba falla, el equipo debe saber inmediatamente de ese hecho, así como informes completos del entorno y las condiciones de prueba en el momento de la falla.

SG VIRTUAL CONFERENCE

Return Of Investment
ROI

Ahorros obtenidos al reemplazar las pruebas de regresión manual con pruebas automatizadas.

$$ROI = \frac{Savings}{Investment}$$

El costo de la inversión en automatización de pruebas.

Fuente: https://www.qaoncloud.com/test-automation-roi-how-do-we-calculate-it/

Return Of Investment
ROI

$$Savings = (tepm - tempa) * No.Test\ Cases * No.Runs$$

$tepm$= Tiempo para ejecutar una prueba manual
$tempa$ = Tiempo para ejecutar la misma prueba en automatización

Fuente: https://www.qaoncloud.com/test-automation-roi-how-do-we-calculate-it/

Return Of Investment
ROI

$$Investment = tcf + (tcta * No.Test\ Cases) + cdm$$

$tcf = Tiempo\ para\ \ construir\ el\ framework$
$tcta = Tiempo\ para\ códicar\ un\ test\ automatizado$
$cdm = Costos\ de\ Mantenimiento$

Fuente: https://www.qaoncloud.com/test-automation-roi-how-do-we-calculate-it/

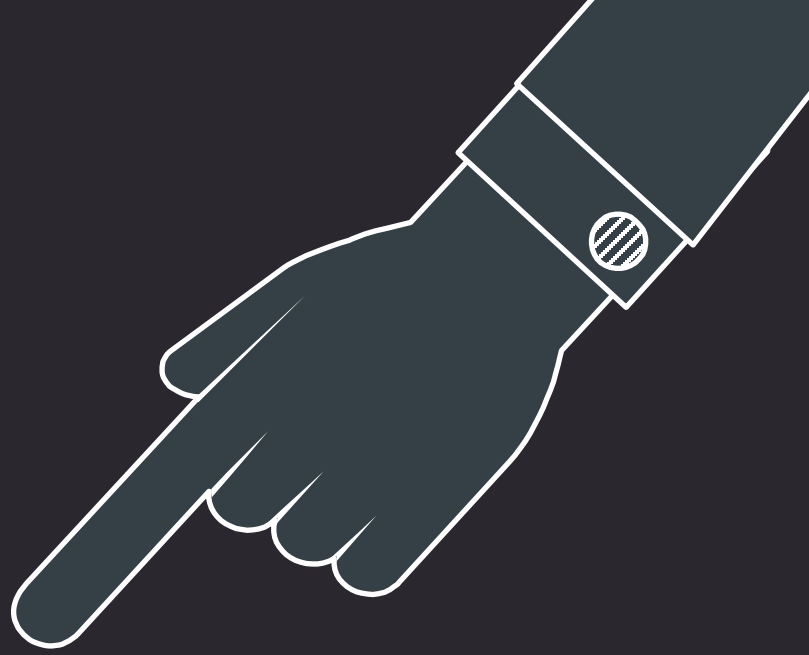
Return Of Investment
ROI

$$Costos\ de\ Mantenimiento = tmcpf * tcfe * No.Test\ Cases * No.Runs$$

$tmcpf = Tiempo\ de\ mantenimiento\ para\ un\ caso\ de\ prueba\ fallido$
$tcfe = \%\ de\ test\ cases\ fallidos\ por\ ejecución$

Fuente: https://www.qaoncloud.com/test-automation-roi-how-do-we-calculate-it/

Return Of Investment
ROI



$$ROI = \frac{(tepm - tempa) * No.Test\ Cases * No.Runs}{tcf + (tcfe * No.Test\ Cases) + (tmcpf * tcta * No.Test\ Cases * No.Runs)}$$

$tcf = Tiempo\ para\ \ construir\ el\ framework$

$tcta = Tiempo\ para\ códicar\ un\ test\ automatizado$

$cdm = Costos\ de\ Mantenimiento$

$tepm = Tiempo\ para\ ejecutar\ una\ prueba\ manual$

$tempa = Tiempo\ para\ ejecutar\ la\ misma\ prueba\ en\ automatización$

$tmcpf = Tiempo\ de\ mantenimiento\ para\ un\ caso\ de\ prueba\ fallido$

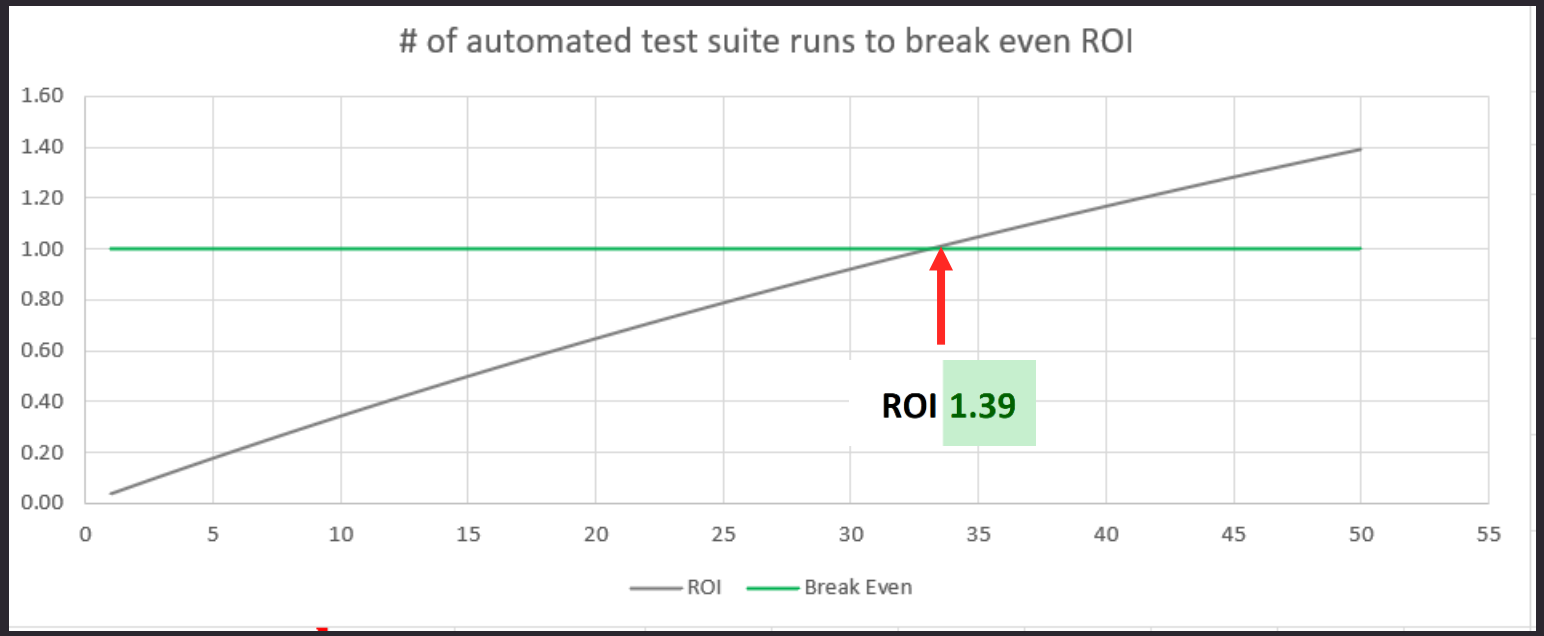
$tcfe = \%\ de\ test\ cases\ fallidos\ por\ ejecución$

Fuente: https://www.qaoncloud.com/test-automation-roi-how-do-we-calculate-it/

Return Of Investment
ROI

Savings		
Avg # of tests per run:	150	tests
# of times tests run:	50	runs
Avg. Manual Test Time (per test):	5	minutes
Avg Automated Test Time (per test):	0.5	minutes

Investment		
Initial Time to build test automation framework:	4	weeks
Avg Time to script automated test:	60	minutes
% of tests requiring maintenance:	5	%
Avg. maintenance Time per failed test:	15	minutes

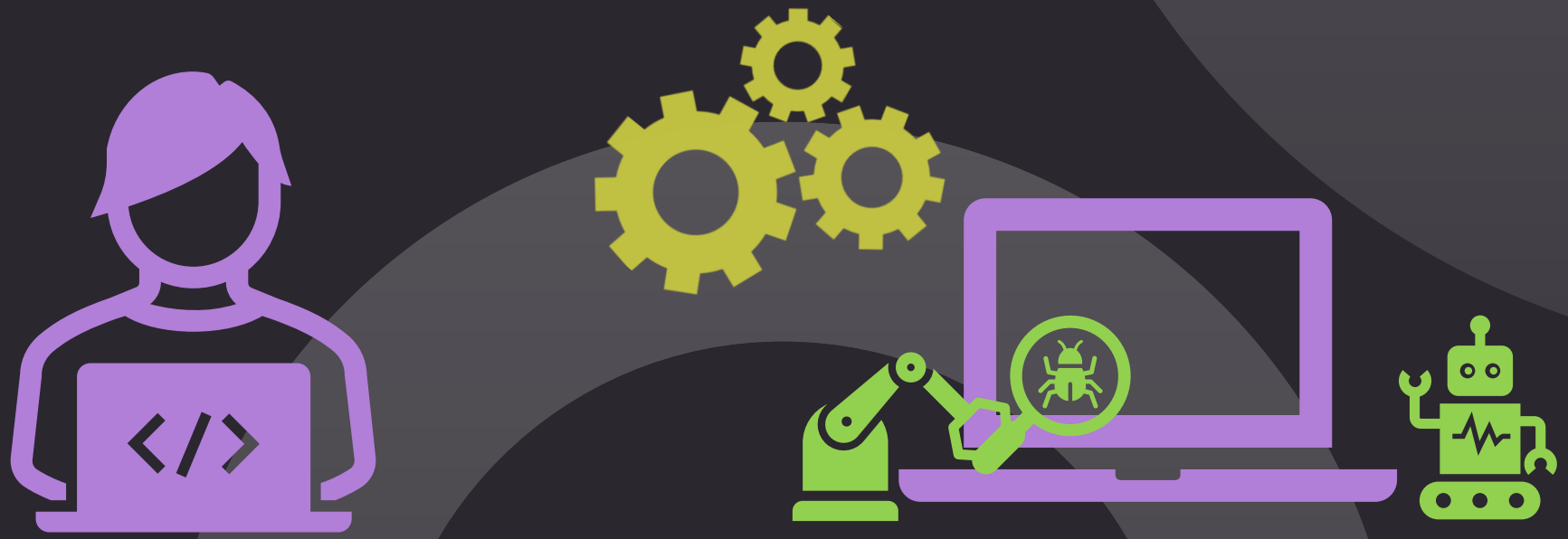


Savings	33750	minutes
Investment	24225	minutes
Net	9525	minutes

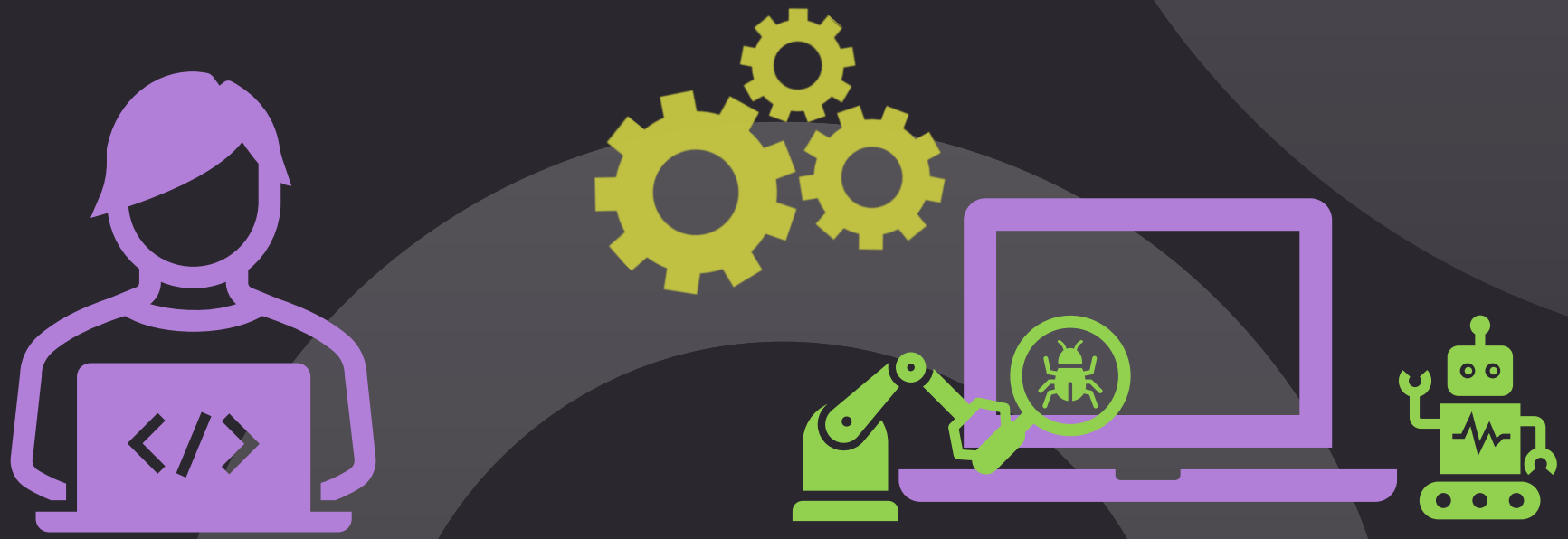
Fuente: https://www.mediafire.com/file/i4rc8lkywyw0fkn/TestAutomationRoiCalculator.xlsx/file

LA ARQUITECTURA

Test Automation Engineer (TAE)

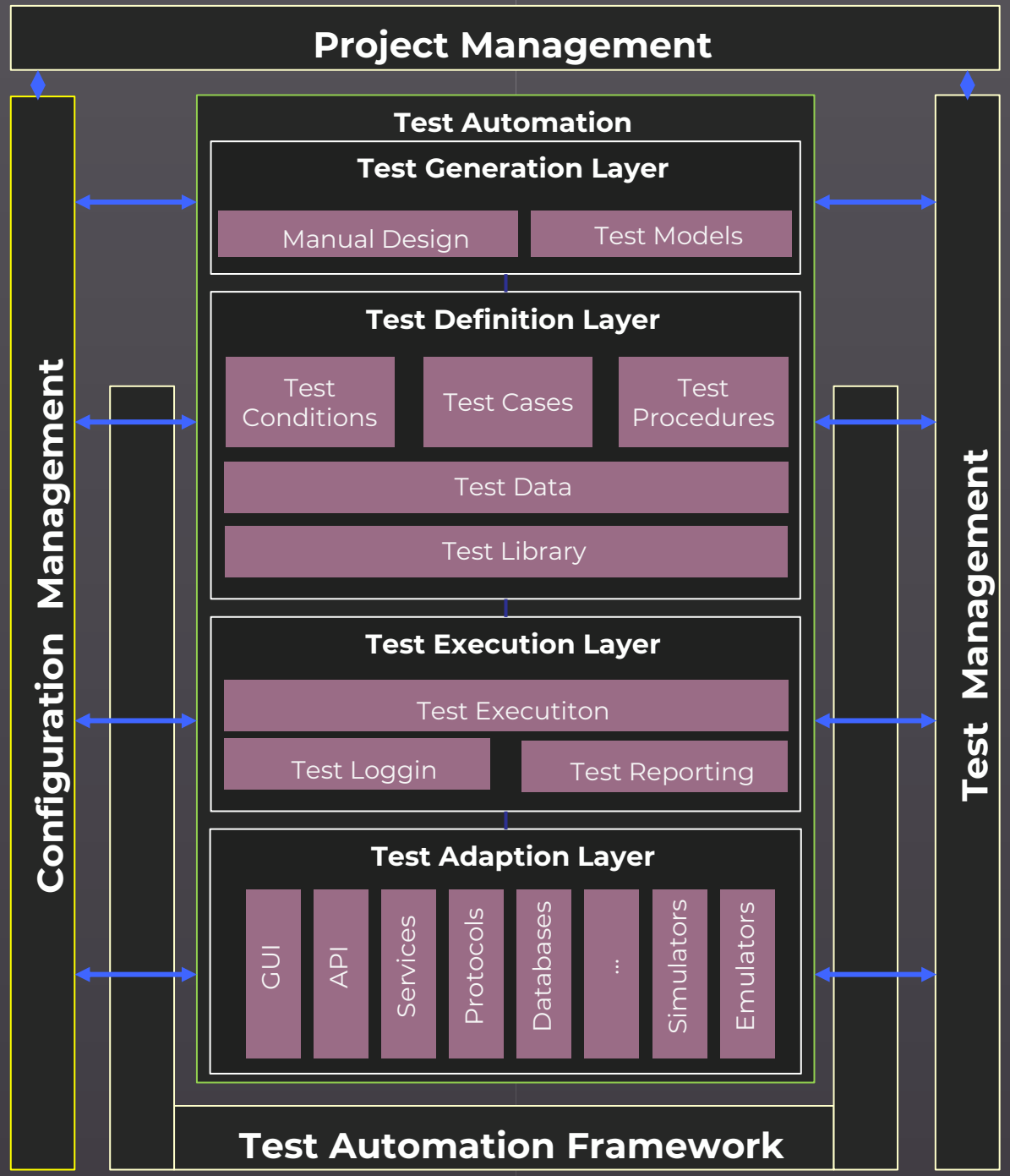


Test Automation Solutions (TAS)



ARQUITECTURA DE AUTOMATIZACIÓN DE PRUEBA GENÉRICA (gTAA)

El gTAA presenta las capas, los componentes y las interfaces, que luego se redefinen en el TAA concreto para un TAS en particular.



Definir el concepto de espacio, capas, servicios e interfaces

Compatibilidad con componentes simplificados

Reutilización de componentes de automatización de prueba

Facilitar el mantenimiento y evolución de los TAS

Definición de las características esenciales para un usuario de un TAS

Definir el concepto de espacio, capas, servicios e interfaces

Compatibilidad con componentes simplificados

Reutilización de componentes de automatización de prueba

Facilitar el mantenimiento y evolución de los TAS

Definición de las características esenciales para un usuario de un TAS

Definir el concepto de espacio, capas, servicios e interfaces

Compatibilidad con componentes simplificados

Reutilización de componentes de automatización de prueba

Facilitar el mantenimiento y evolución de los TAS

Definición de las características esenciales para un usuario de un TAS

Definir el concepto de espacio, capas, servicios e interfaces

Compatibilidad con componentes simplificados

Reutilización de componentes de automatización de prueba

Facilitar el mantenimiento y evolución de los TAS

Definición de las características esenciales para un usuario de un TAS

Definir el concepto de espacio, capas, servicios e interfaces

Compatibilidad con componentes simplificados

Reutilización de componentes de automatización de prueba

Facilitar el mantenimiento y evolución de los TAS

Definición de las características esenciales para un usuario de un TAS

Definir el concepto de espacio, capas, servicios e interfaces

Compatibilidad con componentes simplificados

Reutilización de componentes de automatización de prueba

Facilitar el mantenimiento y evolución de los TAS

Definición de las características esenciales para un usuario de un TAS

Principios para una TAA

S	O	L	I	D
Single Responsibility	Open/Close Extension	Liskov Substitution Replacement	Interface Segregation Component Segregation	Dependency inversion

Enfoques para la Automatización

Capture/Playback

Las entradas al objeto de prueba se registran durante la prueba manual para generar scripts que podrían ejecutarse más tarde (es decir, reproducirse).

Linear Scripting

Una técnica de scripts simple sin ninguna estructura de control en los scripts de prueba.

Structured Scripting

Crea y utiliza una biblioteca de scripts reutilizables (partes de ellas).

Data-Driven Testing

Almacena la entrada de prueba y los resultados esperados en una tabla u hoja de cálculo, de modo que un solo script pueda ejecutar todas las pruebas en la tabla.

Keyword-Driven Testing

Utiliza archivos de datos para contener no solo datos de prueba y resultados esperados, sino también palabras clave relacionadas con el SUT.

Process-Driven Scripting

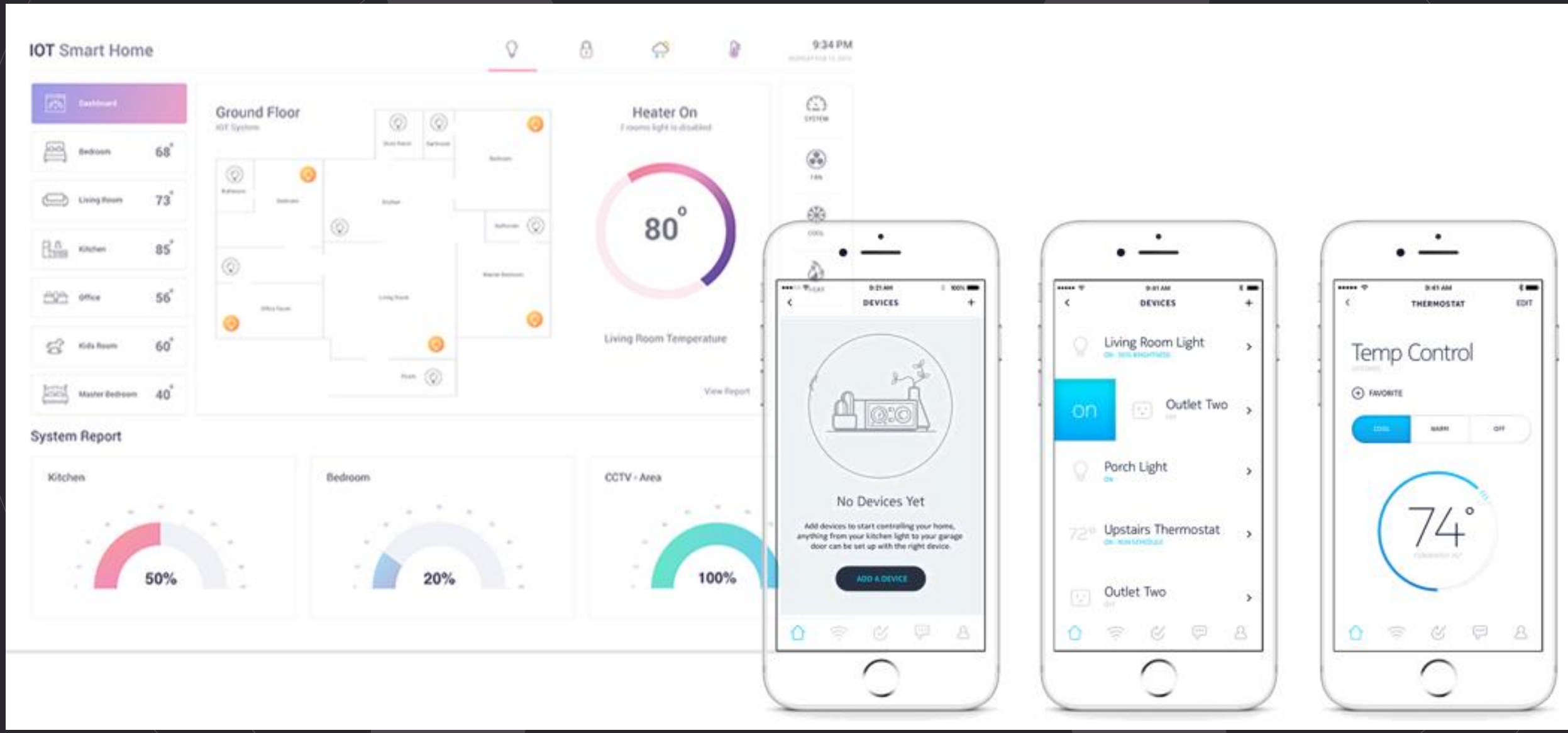
Los scripts se estructuran en escenarios que representan casos de uso del SUT. Estos se pueden parametrizar con datos de prueba.

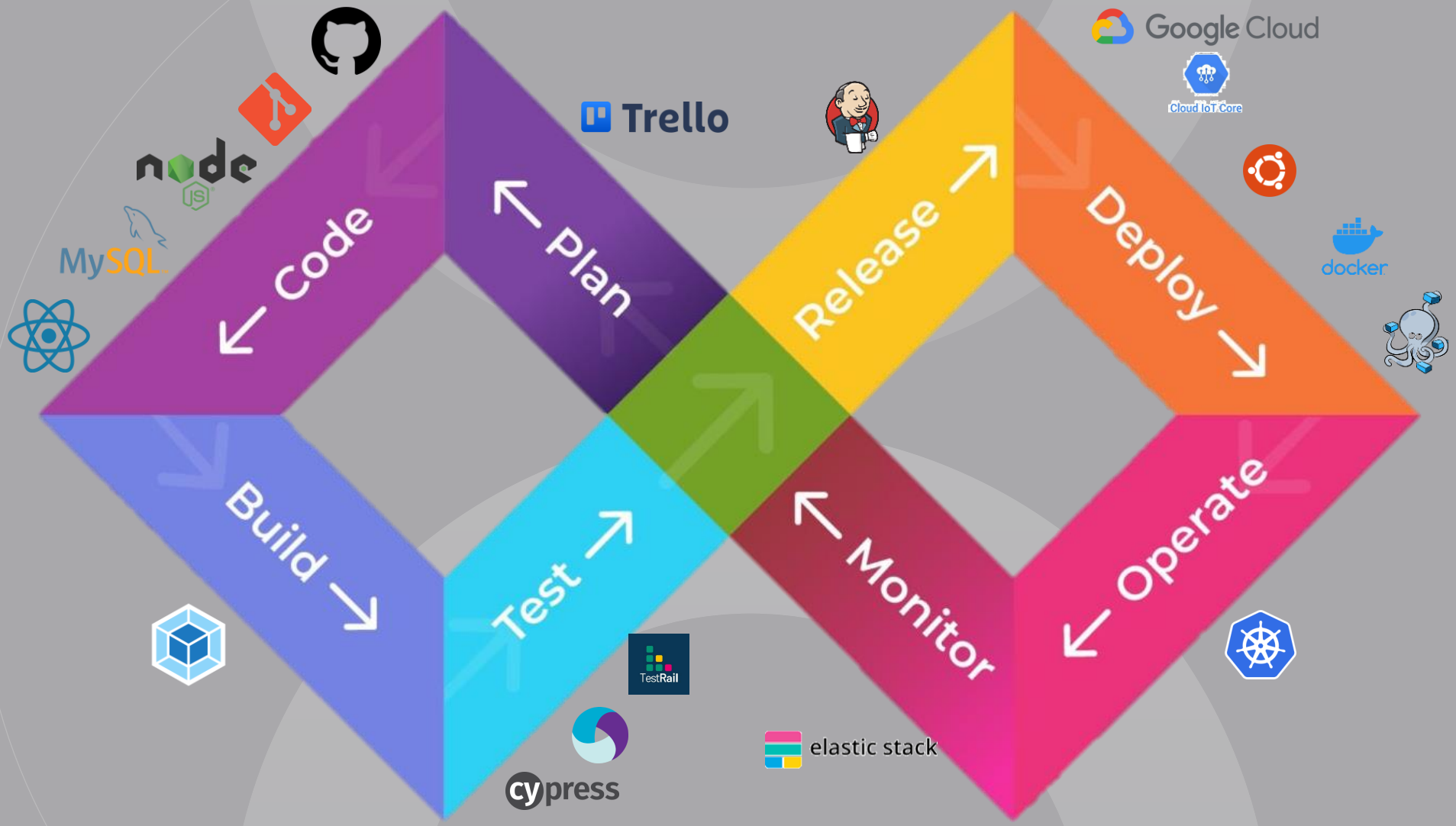
Model-Based testing

la generación automatizada de casos de prueba.

STACK TECNOLÓGICO

Plataformas Web y Mobile




Code
Plan
Release
Deploy
Build
Test
Monitor
Operate
Trello
Google Cloud
Cloud IoT Core
node
MySQL
docker
TestRail
cypress
elastic stack

Lenguajes y Herramientas de Pruebas

Lenguajes






Tools








Test management




Frameworks de Automatización

Linear Automation

Se usa en pruebas de aplicación pequeñas. También se conoce como Record and Play

Library Architecture Testing

Se identifican tareas similares dentro de los scripts y luego se agrupan por función

Data-Driven

Separa los datos de prueba de la lógica del script, lo que significa que los testers pueden almacenar datos externamente

Keyword-Driven

Cada función de la AUT se presenta en una tabla con una serie de instrucciones en orden consecutivo para cada prueba que debe ejecutarse.

Hybrid

Es una combinación de cualquiera de los frameworks mencionados anteriormente que aprovechar sus ventajas y mitiga sus debilidades

SG VIRTUAL CONFERENCE

Frameworks de Automatización

Linear Automation

Se usa en pruebas de aplicación pequeñas. También se conoce como Record and Play

Library Architecture Testing

Se identifican tareas similares dentro de los scripts y luego se agrupan por función

Data-Driven

Separa los datos de prueba de la lógica del script, lo que significa que los testers pueden almacenar datos externamente

Keyword-Driven

Cada función de la AUT se presenta en una tabla con una serie de instrucciones en orden consecutivo para cada prueba que debe ejecutarse.

Hybrid

Es una combinación de cualquiera de los frameworks mencionados anteriormente que aprovechar sus ventajas y mitiga sus debilidades

SG VIRTUAL CONFERENCE

Frameworks de Automatización

Linear Automation

Se usa en pruebas de aplicación pequeñas. También se conoce como Record and Play

Library Architecture Testing

Se identifican tareas similares dentro de los scripts y luego se agrupan por función

Data-Driven

Separa los datos de prueba de la lógica del script, lo que significa que los testers pueden almacenar datos externamente

Keyword-Driven

Cada función de la AUT se presenta en una tabla con una serie de instrucciones en orden consecutivo para cada prueba que debe ejecutarse.

Hybrid

Es una combinación de cualquiera de los frameworks mencionados anteriormente que aprovechar sus ventajas y mitiga sus debilidades

SG VIRTUAL CONFERENCE

Frameworks de Automatización

Linear Automation

Se usa en pruebas de aplicación pequeñas. También se conoce como Record and Play

Library Architecture Testing

Se identifican tareas similares dentro de los scripts y luego se agrupan por función

Data-Driven

Separa los datos de prueba de la lógica del script, lo que significa que los testers pueden almacenar datos externamente

Keyword-Driven

Cada función de la AUT se presenta en una tabla con una serie de instrucciones en orden consecutivo para cada prueba que debe ejecutarse.

Hybrid

Es una combinación de cualquiera de los frameworks mencionados anteriormente que aprovechar sus ventajas y mitiga sus debilidades

Frameworks de Automatización

Linear Automation

Se usa en pruebas de aplicación pequeñas. También se conoce como Record and Play

Library Architecture Testing

Se identifican tareas similares dentro de los scripts y luego se agrupan por función

Data-Driven

Separa los datos de prueba de la lógica del script, lo que significa que los testers pueden almacenar datos externamente

Keyword-Driven

Cada función de la AUT se presenta en una tabla con una serie de instrucciones en orden consecutivo para cada prueba que debe ejecutarse.

Hybrid

Es una combinación de cualquiera de los frameworks mencionados anteriormente que aprovechar sus ventajas y mitiga sus debilidades

SG VIRTUAL CONFERENCE

Patrones de Diseño en Automatización

Driver Factory

Crear una instancia de WebDriver para que las clases de prueba, como los usuarios, no se preocupen si se crean los controladores (drivers)

Page Object Model/Factory

Sirve para crear repositorios de objetos para elementos de interfaz gráfica web

Fluent Interface

Aplicar múltiples propiedades (o métodos) a un objeto conectándolos con puntos (.) Sin tener que volver a especificar el nombre del objeto cada vez

Arrange-Act-Assert

Sugiere dividir una prueba (un método de pruebas) en tres secciones: Inicializar (Arrange), Actuar (Act), Comprobar (Assert)

Page Navigation

Establecer una relación entre cada página, de modo que, en cualquier momento mientras se realiza una operación en un método, puede o no puede (depende) devolver un objeto de página

ScreenPlay

Escribir pruebas de aceptación que se basa en los principios de diseño SOLID

SG VIRTUAL CONFERENCE

Patrones de Diseño en Automatización

Driver Factory

Crear una instancia de WebDriver para que las clases de prueba, como los usuarios, no se preocupen si se crean los controladores (drivers)

Page Object Model/Factory

Sirve para crear repositorios de objetos para elementos de interfaz gráfica web

Fluent Interface

Aplicar múltiples propiedades (o métodos) a un objeto conectándolos con puntos (.) Sin tener que volver a especificar el nombre del objeto cada vez

Arrange-Act-Assert

Sugiere dividir una prueba (un método de pruebas) en tres secciones: Inicializar (Arrange), Actuar (Act), Comprobar (Assert)

Page Navigation

Establecer una relación entre cada página, de modo que, en cualquier momento mientras se realiza una operación en un método, puede o no puede (depende) devolver un objeto de página

ScreenPlay

Escribir pruebas de aceptación que se basa en los principios de diseño SOLID

Patrones de Diseño en Automatización

Driver Factory

Crear una instancia de WebDriver para que las clases de prueba, como los usuarios, no se preocupen si se crean los controladores (drivers)

Page Object Model/Factory

Sirve para crear repositorios de objetos para elementos de interfaz gráfica web

Fluent Interface

Aplicar múltiples propiedades (o métodos) a un objeto conectándolos con puntos (.) Sin tener que volver a especificar el nombre del objeto cada vez

Arrange-Act-Assert

Sugiere dividir una prueba (un método de pruebas) en tres secciones: Inicializar (Arrange), Actuar (Act), Comprobar (Assert)

Page Navigation

Establecer una relación entre cada página, de modo que, en cualquier momento mientras se realiza una operación en un método, puede o no puede (depende) devolver un objeto de página

ScreenPlay

Escribir pruebas de aceptación que se basa en los principios de diseño SOLID

SG VIRTUAL CONFERENCE

Patrones de Diseño en Automatización

Driver Factory

Crear una instancia de WebDriver para que las clases de prueba, como los usuarios, no se preocupen si se crean los controladores (drivers)

Page Object Model/Factory

Sirve para crear repositorios de objetos para elementos de interfaz gráfica web

Fluent Interface

Aplicar múltiples propiedades (o métodos) a un objeto conectándolos con puntos (.) Sin tener que volver a especificar el nombre del objeto cada vez

Arrange-Act-Assert

Sugiere dividir una prueba (un método de pruebas) en tres secciones: Inicializar (Arrange), Actuar (Act), Comprobar (Assert)

Page Navigation

Establecer una relación entre cada página, de modo que, en cualquier momento mientras se realiza una operación en un método, puede o no puede (depende) devolver un objeto de página

ScreenPlay

Escribir pruebas de aceptación que se basa en los principios de diseño SOLID

SG VIRTUAL CONFERENCE

Patrones de Diseño en Automatización

Driver Factory

Crear una instancia de WebDriver para que las clases de prueba, como los usuarios, no se preocupen si se crean los controladores (drivers)

Page Object Model/Factory

Sirve para crear repositorios de objetos para elementos de interfaz gráfica web

Fluent Interface

Aplicar múltiples propiedades (o métodos) a un objeto conectándolos con puntos (.) Sin tener que volver a especificar el nombre del objeto cada vez

Arrange-Act-Assert

Sugiere dividir una prueba (un método de pruebas) en tres secciones: Inicializar (Arrange), Actuar (Act), Comprobar (Assert)

Page Navigation

Establecer una relación entre cada página, de modo que, en cualquier momento mientras se realiza una operación en un método, puede o no puede (depende) devolver un objeto de página

ScreenPlay

Escribir pruebas de aceptación que se basa en los principios de diseño SOLID

SG VIRTUAL CONFERENCE

Patrones de Diseño en Automatización

Driver Factory

Crear una instancia de WebDriver para que las clases de prueba, como los usuarios, no se preocupen si se crean los controladores (drivers)

Page Object Model/Factory

Sirve para crear repositorios de objetos para elementos de interfaz gráfica web

Fluent Interface

Aplicar múltiples propiedades (o métodos) a un objeto conectándolos con puntos (.) Sin tener que volver a especificar el nombre del objeto cada vez

Arrange-Act-Assert

Sugiere dividir una prueba (un método de pruebas) en tres secciones: Inicializar (Arrange), Actuar (Act), Comprobar (Assert)

Page Navigation

Establecer una relación entre cada página, de modo que, en cualquier momento mientras se realiza una operación en un método, puede o no puede (depende) devolver un objeto de página

ScreenPlay

Escribir pruebas de aceptación que se basa en los principios de diseño SOLID

SG VIRTUAL CONFERENCE

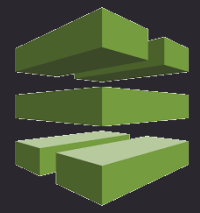
Deployment

Cloud

Control Version

Deploy

Release



MEXI TESTER

SG VIRTUAL CONFERENCE

GRACIAS

¿PREGUNTAS?

SOFTWARE TESTING POR SIEMPRE